z/VM

IBM

TCP/IP Level 3A0 Planning and Customization

Version 3 Release 1.0

SC24-5981-00

z/VM

TCP/IP Level 3A0 Planning and Customization

Version 3 Release 1.0

SC24-5981-00

Note:
Before using this information and the product it supports, read the information under "Notices" on page 631.

First Edition (February 2001)

| This edition applies to the IBM® Transmission Control Protocol/Internet Protocol Feature for z/VM (TCP/IP Level
| 3A0), program number 5654-A17 and to all subsequent releases and modifications until otherwise indicated in new editions.

| This edition replaces SC24-5847-01.

Contents

Preface

TCP/IP Planning and Customization describes how to plan the installation and perform the configuration of the IBM Transmission Control Protocol/Internet Protocol (TCP/IP) Level 3A0 Feature for z/VM.

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

Who Should Read This Book

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

What You Should Know before Reading This Book

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

What This Book Contains

This book describes all applications available with TCP/IP Level 3A0; however, your organization may use only some of these functions.

"Chapter 1. Planning Considerations" on page 1, describes how to plan the installation and customization of TCP/IP Level 3A0.

"Chapter 2. System Requirements for TCP/IP" on page 13, describes the hardware, software, storage, and library requirements needed to install TCP/IP.

"Chapter 3. General TCP/IP Server Configuration" on page 17, describes virtual machine definitions and methods of customizing servers for use with TCP/IP.

"Chapter 4. Configuring the TCP/IP Server" on page 33, describes the syntax of TCP/IP protocol stack configuration statements.

"Chapter 5. Defining the TCP/IP System Parameters" on page 139, describes the syntax for the configuration commands used in the TCPIP DATA file.

"Chapter 6. Configuring the HOSTS LOCAL File" on page 151, describes how to specify local host entries in the TCP/IP host's site table.

"Chapter 7. Configuring the ROUTED Server" on page 155 gives a brief description of the Routing Information Protocol (RIP) and describes the files used in configuring the RouteD server.

| "Chapter 8. Configuring the MPROUTE Server" on page 183, explains the
| MPROUTE's use of the OSPF protocol and configuration parameters.

"Chapter 9. Configuring the DNS Server" on page 223 describes the configuration parameters, how to define the SQL database, and how to install the database.

"Chapter 10. Configuring the BOOTP Server" on page 255, describes how to configure the BOOTPD Virtual Machine.

"Chapter 11. Configuring the DHCP Server" on page 271, describes how to configure the DHCPD Virtual Machine.

"Chapter 12. Configuring the FTP Server" on page 339, describes how to configure the FTP server and how to use the options available when configuring the server.

"Chapter 13. Configuring the Kerberos Server" on page 357, describes how to configure the Kerberos authentication and administration servers, describes the administration functions and commands, and provides a list of the messages produced by the server.

"Chapter 14. Configuring the LPD Server" on page 375, describes how to configure the LPSERVE Virtual Machine.

"Chapter 15. Configuring the RSCS Print Server" on page 387, describes how to configure the RSCS printer server.

"Chapter 16. Configuring the NDB Servers" on page 409, describes how to configure the Network Database (NDB) System.

"Chapter 17. Configuring the NFS Server" on page 413, describes how to configure the VMNFS Virtual Machine.

"Chapter 18. Configuring the Portmapper Server" on page 435, describes how to configure the PORTMAP Virtual Machine.

"Chapter 19. Configuring the REXEC Server" on page 437, describes how to configure the REXECD Virtual Machine.

"Chapter 20. Configuring the SMTP Server" on page 443, describes the syntax of the configuration statements used in setting up the Simple Mail Transfer Protocol (SMTP) server.

"Chapter 21. Configuring the SNALINK Server" on page 505, describes how to configure and operate the SNA LU type 0 Virtual Machine.

"Chapter 22. Configuring the SNMP Servers" on page 513, describes how to configure the SNMPQE and SNMPD Virtual Machines.

| "Chapter 23. Configuring the SSL Server" on page 525, describes how to configure
| the Secure Socket Layer (SSL) server, how to set up the certificate database, and
| how to do certificate database and SSL server administration.

"Chapter 24. Configuring the TFTP Server" on page 547, describes how to configure the TFTPD Virtual Machine.

"Chapter 25. Configuring the UFT Server" on page 565, describes an asynchronous UFT support.

"Chapter 26. Configuring the RSCS UFT Client" on page 579, describes how to configure the UFT server.

"Chapter 27. Configuring the X.25 Interface" on page 583, describes how to configure the X25IPI interface.

"Chapter 28. X Window System Graphical Data Display Manager Support" on page 593, describes how to install the interface for X Window System GDDM® support.

"Chapter 29. Using Translation Tables" on page 595, describes how to customize the translation tables supplied with TCP/IP.

"Chapter 30. Testing" on page 607, describes how to test and debug TCP/IP.

"Chapter 31. Using Source Code Libraries" on page 611, describes how to maintain the source code libraries for TCP/IP.

"Appendix A. Using TCP/IP with an External Security Manager" on page 617, shows how RACF® offers effective user verification, resource authorization, and logging capabilities for TCP/IP.

"Appendix B. Related Protocol Specifications" on page 623, contains a list of some of the Related Protocol Specifications used in TCP/IP.

"Appendix C. Abbreviations and Acronyms" on page 627, lists the abbreviations and acronyms that are used throughout this book.

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

How to Use This Book

Read the first chapters to plan, install, and configure the TCPIP virtual machine. Read the middle chapters if you plan to configure any other servers or virtual machine to run with TCP/IP. Read the last few chapters to learn about translation tables, testing, source code libraries, and GDDM.

How the Term "internet" Is Used in This Book

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

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

Understanding Syntax Diagrams

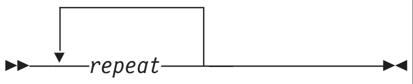
This section describes how to read the syntax diagrams in this book.

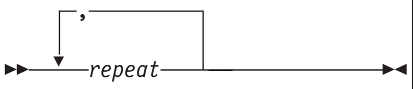
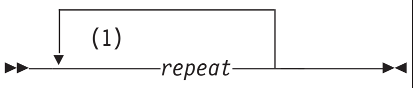
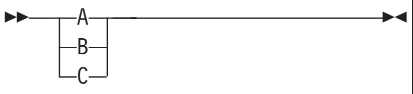
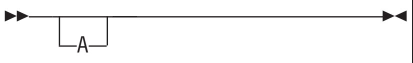
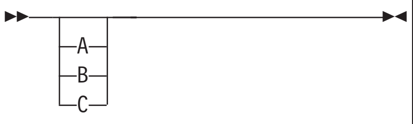
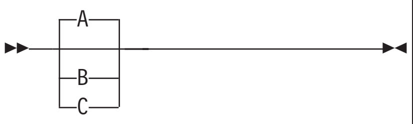
Getting Started: To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

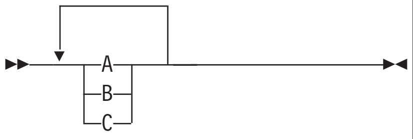
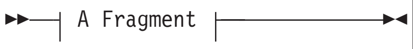
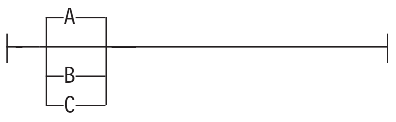
- The ►►── symbol indicates the beginning of a syntax diagram.
- The ──► symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ►── symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.

- The ───►◄ symbol indicates the end of a syntax diagram.

Syntax items (for example, a keyword or variable) may be:
- Directly on the line (required)
- Above the line (default)
- Below the line (optional).

Syntax Diagram Description	Example
Abbreviations: Uppercase letters denote the shortest acceptable abbreviation. If an item appears entirely in uppercase letters, it cannot be abbreviated. You can type the item in uppercase letters, lowercase letters, or any combination. In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.	`►►──KEYWOrd──────────────────►◄`
Symbols: You must code these symbols exactly as they appear in the syntax diagram.	***** Asterisk **:** Colon **,** Comma **=** Equal Sign **-** Hyphen **()** Parentheses **.** Period
Variables: Highlighted lowercase items (*like this*) denote variables. In this example, *var_name* represents a variable you must specify when you code the KEYWORD command.	`►►──KEYWOrd──`*`var_name`*`──────────►◄`
Repetition: An arrow returning to the left means that the item can be repeated.	

Syntax Diagram Description	Example
A character within the arrow means you must separate repeated items with that character.	
A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.	 **Notes:** **1** Specify *repeat* up to 5 times.
Required Choices: When two or more items are in a stack and one of them is on the line, you *must* specify one item. In this example, you must choose A, B, or C.	
Optional Choice: When an item is below the line, the item is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	 
Defaults: Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	

Syntax Diagram Description	Example
Repeatable Choices: A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	
Syntax Fragments: Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	 **A Fragment:** 

Where to Find More Information

"Appendix C. Abbreviations and Acronyms" on page 627, lists the abbreviations and acronyms that are used throughout this book.

The "Glossary" on page 635, defines terms used throughout this book that are associated with TCP/IP communication in an internet environment.

For more information about related publications, see "Bibliography" on page 653.

How to Send Your Comments to IBM

Your feedback is important in helping us to provide the most accurate and high-quality information. If you have comments about this book or any other VM documentation, send your comments to us using one of the following methods. Be sure to include the name of the book, the form number (including the suffix), and the page, section title, or topic you are commenting on.

- Visit the z/VM web site at:

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

  There you will find the feedback page where you can enter and submit your comments.
- Send your comments by electronic mail to one of the following addresses:

Internet:	pubrcf@vnet.ibm.com
IBMLink™:	GDLVME(PUBRCF)

- Fill out the Readers' Comments form at the back of this book and return it using one of the following methods:
  - Mail it to the address printed on the form (no postage required in the USA).
  - Fax it to 1-607-752-2327.

– Give it to an IBM representative.

Summary of Changes

This section describes the technical changes made in this edition of the book and in previous editions. For your convenience, the changes made in this edition are identified in the text by a vertical bar (|) in the left margin. This edition may also include minor corrections and editorial changes that are not identified.

First Edition for z/VM (February 2001)

This edition contains updates for the General Availability of z/VM 3.1.0.

Native Queued Direct I/O Support

Native Queued Direct I/O support allows TCP/IP to support the microcode that provides the high speed GigaBit Ethernet Adapter, which utilizes the Queued Direct I/O Hardware Facility (QDIO). The following have been added:

- A DEVICE statement, used to specify the name and address of the device that will use the QDIO facility.
- The FIXEDPAGESTORAGEPOOL configuration statement that is used to set the initial number of Fixed Page Storage Blocks.
- Several new Process names that are valid for the TRACE, LESSTRACE, MORETRACE, and NOTRACE statements.

Open Shortest Path First (OSPF) Protocol

A new chapter has been added, ″Configuring the MRPOUTE Server″. This chapter describes how to configure OMPROUTE, and also describes the implementation of the Open Shortest Path First (OSPF) protocol. This chapter also contains several new OSPF and RIP configuration statements, descriptions, and parameters.

The ″Using OSPF″ section describes the protocol features, as well as supported network types. It details the setting of OSPF interface types and required tasks on configuring the OSPF protocol.

IP Multicast Support

IP Multicast support provides the ability to send an IP datagram to a group of hosts that are members of a single multicast group, and the ability for members of a multicast group to receive multicast datagrams. The ability to specify multicast options on sockets is also available. The following changes include:

- A new usage note added to the GATEWAY statement enabling multicast routes to be specified using a host syntax
- A new usage note added to the TRANSLATE statement enabling a token ring link to be configured to broadcast multicast datagrams instead of using the functional MAC address
- Updates to the *dest_addr* operand, and updates and additions to the usage notes of the BSDROUTINGPARMS statement
- Updates to a usage note of the OBEYFILE command
- Updates to RouteD to handle IP multicasting for RIPv2 packets. Four new operands were added to the ROUTED config statement for SMSG support: CONFIG, RECONFIG, RGATEWAYS, and DGATEWAYS.

Secure Socket Layer (SSL) Support

SSL support provides secure (encrypted) communication between a remote client and a VM TCP/IP server. Any VM server that listens on a secure port can participate in an SSL connection with any external client that supports SSL according to RFC 2246. Under SSL protocol, the server is always authenticated and must provide a certificate to prove its identity. An installation identifies the ports that are secure and obtains the certificates to be used, but no changes to VM servers are required to use the SSL support.

The processing for SSL support is provided by a new SSL server. The SSL server manages the certificate database and handles the encryption and decryption of data. The PORT statement has been enhanced to allow the designation of secure ports. A new SSLADMIN command interface provides certificate database and SSL server administration functions.

VM NFS Client Support

VM NFS Client support allows BFS users and applications transparent access to data on remote systems with NFS servers supporting SUN NFS V2 or V3 protocols, or both. With the necessary NFS server support, the VM NFS Client can access the data on that system.

The remote data accessed by an NFS client is mounted on the CMS BFS structure in a single virtual machine. A mount is issued to CMS to attach the remote file system to mount point on the CMS BFS structure. Once mounted, the remote file system is accessible using interfaces currently available on z/VM. This new function has added the following:

- A VMFILETYPE statement, used to specify the translation and line values for a file extension (file type)
- A VMFILETYPEDEFAULT statement, used to specify the translation and line values for file extensions (file types) that do not have VMFILETYPE statements.

See the *z/VM: OpenExtensions Command Reference* for more information about the VM NFS Client.

FTP Web Browser Support

The FTP server has been enhanced to better accommodate web browser and graphical FTP clients. The changes and new function include:

- The ability to provide UNIX®-format list information in response to client DIR subcommand requests
- The ability to perform automatic EBCDIC-ASCII data translation for transferred files, based on file types (or, file extensions)
- The FTP server now makes use of a server-specific configuration file for its startup information, so that configuration is more consistent with other TCP/IP servers.

Other Changes

- The chapter ″Configuring the Standard X Client″ was removed; support is no longer provided for this.
- The chapter ″Configuring the NCS Servers″ was removed; support is no longer provided for this.

Second Edition for VM/ESA® (July 1999)

This edition contains updates for the General Availability of VM/ESA 2.4.0.

RouteD Support

RouteD has been updated to provide the following enhancements:

- RIP Version 2 - This is an extension of the RIP Version 1 protocol.
- Compatibility with OS/390® RouteD
- Variable Subnetting
- Virtual IP Addressing (VIPA)

Native ATM Support

Native ATM support has been added, which enables TCP/IP to use an Open Systems Adapter configured to support Native ATM.

FTP Server Enhancements

The following enhancements have been made to the FTP Server:

- The VM Special Message Facility (SMSG) is available to provide an interface to communicate with the VM FTP server virtual machine.
- Support has been added for an FTP welcome banner, which displays a site-spec message when connecting to the FTP server or displays a user-specific message when a login occurs.
- A new start-up parameter will enable FTP reader file support, allowing users to STOR files to the virtual reader of a VM user ID.
- New start-up parameters will enable the FTP User Exits.

SMTP Server Enhancements

The following enhancements have been made to the SMTP server:

- The processing framework for SMTP service extensions has been added; this framework includes support for the 8BITMIME and SIZE service extensions.
- New SMTP configuration statements, 8BITMIME and SOURCEROUTES, are supported to facilitate 8-bit MIME support and mail source route handling.

VMNFS Server Enhancements

The following enhancements have been made for the VMNFS server machine:

- Several new records have been added, which can be defined in the VMNFS CONFIG file for configuring the Network File System: DUMPMOUNT, EXPORT, EXPORTON MAXTCPUSERS, PCNFSD, and VMFILETYPE VMFILETYPE
- The VM Special Message Facility (SMSG) is available to provide an interface to communicate with the VM NFS service machine.
- New start-up parameters allow you to restrict the server to the Version 2 NFS level and UDP communications.
- You can configure whether mounts should be restricted to exported file systems only, and whether or not the server should provide a list of mounted file systems.
- The new exit VMNFSSMG was added, which screens all SMSG requests.
- The VMNFSMON exit was improved to handle new MOUNT options.

Other Changes

- New DOMAINLOOKUP and DOMAINSEARCH configuration statements have been added to the TCPIP DATA file to support more flexible host name resolution processing.
- ATMARPSERVER, ATMLIS, and ATMPVD statements, and ATM-specific device and link statements are now supported for TCP/IP server configuration.
- The Telnet Session Connection Exit and Telnet Printer Management Exit appendices were moved from this book into the *z/VM: TCP/IP Level 3A0 Programmer's Reference*.
- Chapter 25, "Configuring the RSCS UFT Client" is a new chapter that describe how to configure an asynchronous UFT client.
- Chapters 3, 4, and 5 from TCP/IP Function Level 310 have been consolidated into a new chapter.

First Edition for VM/ESA

This edition contains updates for the General Availability of VM/ESA 2.3.0.

Changes

- The following new statements have been added:
  - ATSIGN, SMTPSERVERID, and UFTSERVERID for defining TCP/IP system parameters.
  - VERIFYCLIENT, VERIFYCLIENTDELAY, and TRACE for the SMTP server configuration file.
  - DATABUFFERLIMITS, MONITORRECORDS, TN3270E, and TRACEONLY for the TCPIP server configuration file.
  - FAILEDJOB for the LPD server configuration file.
- The PORT statement has been extended to permit reservations to be made for particular home addresses.
- The HOME statement's method of handling multiple IP addresses for the same link has changed.
- Several new parameters have been added to the INTERNALCLIENTPARMS statement.
- The following changes have been made for the VMNFS server machine:
  - The VMNFS CONFIG file allows you to configure your VMNFS server machine. You can define the values used for file extension translation, and you can define whether or not PCNFSD is supported.
  - More detailed tracing is available with the use of the MASK start-up or SMSG command. Subnetting of trace information is also provided.
  - ANONYMOUS requests to mount BFS and SFS directories can be allowed or disallowed.
- The following sample configuration files have been updated:
  - PROFILE TCPIP
  - SMTP CONFIG
  - TCPIP DATA
- The following new virtual machines have been added:
  - BOOTPD
  - DHCPD
  - TFTPD

  – UFTD

- Chapter 5, ″Methods of Server Configuration″, was rewritten to include a newly provided simplified method for customizing and configuring TCP/IP servers, which uses a file with a Names file format.
- Chapter 6, ″Configuring the TCPIP Virtual Machine″, was rewritten to be clearer, easier to read, and more task-oriented.
- Chapter 12, ″Printing with the RSCS Server″ is a new chapter that provides an enhanced level of TCP/IP print support, including LPR, LPD and TN3270E.
- Chapter 14, ″Configuring the UFTD Virtual Machine″ is a new chapter that describes how to configure the UFTD server.
- Chapter 26, ″Configuring the BOOTPD Virtual Machine″ is a new chapter that describes how to configure the BOOTPD server. The Bootstrap Protocol Daemon responds to client requests for boot information using information contained in a BOOTP machine file.
- Chapter 27, ″Configuring the TFTPD Virtual Machine″ is a new chapter that describes how to configure the TFTPD server. The Trivial File Transfer Protocol Daemon transfers files between the Byte File System (BFS) and TFTP clients. TFTPD supports access to files maintained in a BFS directory structure that is mounted.
- Chapter 28, ″Configuring the DHCPD Virtual Machine″ is a new chapter that describes how to configure the DHCPD server. The Dynamic Host Configuration Protocol Daemon server responds to client requests for boot information using information contained in a DHCP machine, which includes the IP address of the client, the IP address of the TFTP daemon, and file information to request from the TFTP daemon.
- Miscellaneous service APARs were added since the previous release.

Chapter 1. Planning Considerations

This chapter provides an introduction to the Transmission Control Protocol/Internet Protocol, TCP/IP, and describes the planning and preparation that you should consider before TCP/IP Level 3A0 is on your system.

Introducing TCP/IP

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

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

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

Connectivity and Gateway Functions

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

- **X.25** - An interface consisting of a data terminal equipment (DTE) and a data circuit-terminating equipment (DCE) in communication over a link using the procedures described in the CITT Recommendation X.25.
- **Point-to-Point** - The Point-to-Point function of TCP/IP serves three main components: a method for encapsulating datagrams over serial lines; an extensible Link Control Protocol to configure and test data-link connections; and a method for establishing different network-layer protocols or Network Control Protocols (NCP). They include CTC and IUCV connections.
- **SNALINK** - A function of TCP/IP products for VM and MVS that allows the use of an SNA routing network to transfer data using the TCP/IP protocols. Data can be received, sent and routed to other systems using SNALINK.
- **Token-Ring Network** - A ring network that allows unidirectional data transmission between data stations. Using a token passing procedure, data is sent and returned to the transmitting station.
- **Ethernet Network** - A 10-Mbps baseband local area network that allows multiple stations to access the transmission medium at will without prior coordination,

avoids contention by using carrier sense and deference, and resolves contention by using collision detection and transmission. Ethernet uses carrier sense multiple access with collision detection.

- **Fiber Distributed Data Interface** - TCP/IP for VM supports Fiber Distributed Data Interface (FDDI) communications. There are two types of connections that can be defined. Dual Attached Stations (Class A) which are connected to both a primary and secondary ring and are capable of redirecting traffic from one ring to the other. The second, Single Attached Stations (Class B) which are connected to a single ring and are attached to a Dual Attach Concentrator. FDDI is also know as IEEE 802.5.
- **ATM** - IP over native ATM enables TCP/IP to send data across an ATM network over ATM virtual circuits. Support extends beyond LAN emulation; native definition of ATM addresses does not require definition of LAN addresses. VM TCP/IP uses the IBM S/390® Open Systems Adapter (OSA-2) to enable this support. IP data over both "best effort" Switched Virtual Circuits (SVCs) and "best effort" Permanent Virtual Circuits (PVCs) is supported.

Server Functions

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

- **BOOTPD** - The BOOTP virtual machine (daemon) responds to client requests for boot information using information contained in a BOOTP machine file.
- **DHCPD** - The DHCP daemon (DHCPD server) responds to client requests for boot information using information contained in a DHCP machine file. This information includes the IP address of the client, the IP address of the TFTP daemon and information about the files to request from the TFTP daemon.
- **FTP** - The File Transfer Protocol virtual machine (FTPSERVE) serves client requests from the TCPIP virtual machine through VMCF communication. The FTP server allows you to transfer files between your local host and a foreign host that supports TCP/IP. When invoked, FTP establishes a connection to a foreign host's FTP server. After you have identified yourself to the foreign FTP server, you can retrieve information about the foreign server, list files, transfer files, delete files, rename files and execute CMS commands.
- **LPD** - The Line Printer Daemon virtual machine (LPSERVE) serves client requests to print a file. The LPSERVE server looks for the printer specified by the client (LPR) and attempts to print the document. The printer itself is not dedicated to TCP/IP, it may receive output from other sources as well. The LPSERVE server supports three types of printer connections:
  - Local Printers
  - Remote RSCS printers (RSCS connected)
  - Remote TCP/IP printers (TCP/IP connected)

  The local printers are physically attached to the VM system. Any printer available through the RSCS network however, can be accessed with the remote RSCS printer support. This printer might be attached to an OS/390 system without a TCP/IP connection. Any printer controlled by another LPD server can be made available to a VM TCP/IP user.
- **NAMESRV** - The Domain Name Server (NAMESRV) serves to resolve domain names, aliases for clients and provides commands for querying name servers from a CMS user ID. Name servers manage two kinds of data which can be stored in either local data files (HOSTS LOCAL) or in DB2® Server for VSE & VM (formerly known as SQL/DS™ databases). The first kind of data is held in sets called *zones*; each zone is the complete database. The second kind of data is cached data, which is acquired by a local resolver.

- **NDB** - The Network Database server uses the Remote Procedure Call (RPC) protocol to allow interoperability among a variety of workstation users and a host relational database system. It provides access to a mainframe relational database from workstations and allows users to issue SQL statements interactively, or embed SQL statements within an application program.
- **NFS** - The Network File System virtual machine (VMNFS) allows a client system to access CMS minidisks, SFS directories, and BFS directories as if they were local to the client. VMNFS allows users to execute programs, and to create and edit files. The VMNFS virtual machine requires access to the IBM Language Environment® runtime library during execution.
- **REXEC** - The Remote Execution virtual machine (REXECD) sends a single command to a remote system for execution. The remote system name, user ID, password and command string can all be passed as parameters. The REXECD server can log on, execute the command and log off. REXECD can also be used for network management purposes. Procedures to start and stop TCP/IP devices (physical links) can be initialized using the REXEC command and the OBEYFILE command.
- **SMTP** - The Simple Mail Transfer Protocol virtual machine (SMTP) operates as a mail gateway between TCP/IP network sites and RSCS network sites. This means, for example, OfficeVision® users can exchange mail with UNIX® workstations through the VM TCP/IP SMTP gateway. The SMTP server allows you to create custom mail headers (RFC822), provides an interface which allows the querying of SMTP mail delivery queues and provides a set of privileged commands for system administrator tasks. SMTP can be configured to use either a domain name server or the local site tables.
- **SSL** - The Secure Socket Layer (SSL) server, which runs in the SSLSERV virtual machine, provides processing support for secure (encrypted) communication between remote clients and VM TCP/IP servers that listen on secure ports. The SSL server manages the database in which the server authentication certificates are stored. The TCP/IP (stack) server routes requests for secure ports to the SSL server. The SSL server, representing the requested application server, participates in the handshake with the client in which the cryptographic parameters are established for the session. The SSL server then handles all the encryption and decryption of data.
- **TELNET** - The Teletypewriter Network (TELNET) feature is part of the TCPIP virtual machine. The TELNET protocol provides a standardized interface, through which a program on one host (the TELNET client) may access the resources of another host (the TELNET server) as though the client were a local terminal connected to the server. TELNET provides interactive access to local and remote hosts, support for TN3270 and TN3270E, but does not support graphics.
- **TFTPD** - The TFTP daemon (TFTPD server) transfers files between the Byte File System (BFS) and TFTP clients. TFTPD supports access to files maintained in a BFS directory structure that is mounted.
- **UFTD** - The UFT daemon (UFTD server) accepts unsolicited file transfer requests to receive files from remote clients for local users.
- **Port Mapper** - The Port Mapper (PORTMAP) server application is used to map program numbers and various numbers to RPC programs that request information. The PORTMAP server only knows about RPC programs on the local host system. The calling application contacts PORTMAP on the destination host to obtain the correct port number of a specific remote program.

Client Functions

- **FTP** - The FTP client sends requests to the FTP server virtual machine.
- **LPR** - the LPR command is used to communicate with the LPSERVE server to submit commands and documents to be printed.
- **NOTE** - The NOTE command (provided with CMS) allows users to send mail throughout the local network and to external network sites.
- **NSLOOKUP** - The NSLOOKUP command is used to query names and allows you to locate information about network nodes, examine the content of a name server database and establish the accessibility of name servers.
- **REXEC** - The REXEC client in VM/CMS issues requests to execute commands on other network systems.
- **SENDFILE** - The SENDFILE command (provided with CMS) allows users to send files throughout the local network and to external network sites.
- **TELNET** - The TELNET client allows you to log on from any VM/CMS terminal to either local or remote networks that are operating TCP/IP.
- **TFTP** - The TFTP application is used to transfer files among personal computers. TFTP allows files to be sent and received but does not provide any password protection or directory capability.

Network Status/Management Functions

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

  The Open Shortest Path first protocol (OSPF) is an Interior Gateway Protocol. It distributes routing information between routers that belong to a single autonomous system; that is, a group of routers that all use a common routing protocol.
- **ROUTED** - The Routing Information Protocol (RIP) creates and maintains network routing tables. RIP arranges to have gateways and routers periodically broadcast their routing tables to neighbors. This protocol is most useful as an Interior Gateway Protocol.
- **SNMP** - The Simple Network Management Protocol (SNMP) virtual machine allows you to manage your TCP/IP network. SNMP is an internet standard that defines a set of functions that may be used to monitor and control network elements. The SNMP server (or agent) responds to commands issued by the client (or monitor).

Application Programming Interfaces

- **KERBEROS** - The Kerberos Authentication and Authorization system is an encryption-based security system that provides mutual authentication between users and servers in a network environment. The VM Kerberos System relies upon two virtual machines. VMKERB which runs the authentication server and ADMSERV which runs the Kerberos database remote administration server.

- **RPC** - The Remote Procedure Call is an application programming interface (API) for developing distributed applications. It allows programs to call subroutines that are executed at a remote system. The caller program (client) sends a call message to the server, and waits for a reply message. In order to send a message the caller must know the exact port number used by the RPC program. To get this port number, the caller sends a request to the PORTMAP server on the remote host to retrieve the port information for the desired program.
- **SNMP DPI** - The Simple Network Management Protocol Daemon (SNMPD) is a server that communicates management information between network management stations and agents in the network. The ***SNMP Distributed Programming Interface (DPI)*** allows users to manipulate the information or MIB variables for each layer in the TCP/IP protocol.
- **Sockets** - The Sockets API provides a simplified procedure to transfer information and to communicate between applications. The socket interface is designed to provide applications with a network interface that hides the details of the physical network.
- **X Window System** - The X WINDOW SYSTEM is one of the most widely used ***Graphical User Interfaces (GUI)*** or bitmapped- window display systems. It is supported by all major workstation vendors and is used by thousands of users worldwide. An interface, Graphical Data Display Manager (GDDM®), is provided and permits graphics display output from the IBM GDDM to be displayed on workstations that support the X Windows™ System.

Migration Considerations

In addition to the introduction of new TCP/IP functions, there have been changes to some existing functions. In a few cases, functions have been deleted or replaced by alternative functions. This section describes these changes or deletions.

TCP/IP and related features are supported on associated releases of CP and CMS only.

Packaging

General Information about TCP/IP Level 3A0

- TCP/IP is included as a priced feature of the z/VM product, requiring that you have a license from IBM for its use. The TCP/IP feature **must be enabled**, using the correct feature code, before it can be used.
- | The Network File System (NFS) server is a separately licensed and priced feature of z/VM. Like the base TCP/IP feature, it is included with the z/VM product, but **must be enabled** before it can be used.

  | For information about enabling the TCP/IP and NFS server features, see the *TCP/IP Feature for z/VM, Level 3A0 Program Directory*.
- This level of TCP/IP is not separately orderable, though it is serviced separately from the z/VM product.
- This level of TCP/IP relies on the presence of certain functions in CP and CMS. Abends and incorrect results are possible if you attempt to use it with an older level of CP or CMS.
- TCP/IP softcopy publications are included with the other z/VM softcopy publications.

Changes Introduced in TCP/IP Level 3A0

- DES Kerberos functions are now incorporated in the base TCP/IP feature (non-DES Kerberos functions are no longer available). Therefore, customers who have a Kerberos database that was created in a non-DES environment **must** delete and then rebuild that database using the supplied DES Kerberos functions. Refer to "Chapter 13. Configuring the Kerberos Server" on page 357 for more information about building the Kerberos database.
- The various servers and code that provide support for the Network Computing System (NCS) are no longer provided.
- The installation user ID-owned "Sample" (2C2) minidisk and its corresponding sample files are no longer provided. The sample files provided on this disk in prior levels of TCP/IP can be obtained via the IBM TCP/IP for VM Internet home page; the URL is:

  `http://www.ibm.com/s390/vm/related/tcpip/`

Changes Introduced in TCP/IP Function Level 310

- Support for TCP/IP-based e-mail is included in CMS. The TCP/IP versions of NOTE and SENDFILE are no longer provided.
- A subset of RSCS Version 3 Release 2 function is available with the z/VM product to provide enhanced TCP/IP client and server print capabilities. RSCS functions not related to TCP/IP printing enhancements require a separate RSCS license for use. Refer to the *RSCS Version 3 Release 2 Program Directory* for additional information.
- Non-DES Kerberos functions are included in the base TCP/IP Feature. DES Kerberos functions are available as a separate feature that must be separately installed.

Server Configuration

Changes Introduced in TCP/IP Function Level 310

- Separate server initialization execs (TCPIPXIT, FTPDEXIT, etc.) are no longer used. All server parameters and features are controlled by the DTCPARMS file. There is support for you to supply an exit specific to a particular server, or an exit that is used by all servers.
- Changing server names (when defining a second set of TCP/IP servers, for example) no longer requires changes to IBM-provided execs. All information related to the user ID of a particular server is kept in the TCPIP DATA file, or is part of the server configuration and is kept in the DTCPARMS file.
- External Security Manager (ESM) interfaces have been standardized across all servers. The RPIUCMS command will be used to initialize the RACROUTE environment, and RPIVAL will be used to validate user IDs and passwords supplied by clients. These commands can be changed using a customized SYSTEM DTCPARMS file.
- The ESM environment is automatically established for a server when `:ESM_Enable.YES` is specified for that server in the SYSTEM DTCPARMS file.

Refer to "Chapter 3. General TCP/IP Server Configuration" on page 17 for more information.

TCPIP Server

Changes Introduced in TCP/IP Function Level 320

- The **TIMESTAMP** statement default has been changed from TIMESTAMP 0 to TIMESTAMP PREFIX, which specifies that a time stamp will prefix every trace and console message. This change helps in diagnosing problems and isolating error conditions. Therefore, it is recommended that any existing TCPIP configuration files be changed to specify TIMESTAMP PREFIX to aid in problem determination.
- The Telnet session connection exit interface has been changed to pass the LU name supplied by a client, if any, as an additional parameter. Existing exits may need to be changed to accommodate this behavior.
- The Telnet printer management exit is called for any printer session, regardless of whether the client LU name and IP address are defined by a TN3270E statement in the TCP/IP configuration file.

RouteD/MPROUTE Server

Changes Introduced in TCP/IP Level 3A0

- The RouteD and MPROUTE servers cannot be concurrently used with the same TCP/IP stack server.

Changes Introduced in TCP/IP Function Level 320

- The **#CP EXTERNAL** command is no longer supported by the RouteD server. The command to stop RouteD is **#CP SMSG * SHUTDOWN**.

DNS Server

Changes Introduced in TCP/IP Level 3A0

The use of a cache file (as used with a CACHINGONLY name server configuration) has been expanded to the PRIMARY and SECONDARY configurations. A new ROOTCACHE statement allows a cache file to be specified in a manner similar to that specified on the CACHINGONLY statement. A corresponding sample root cache file, NSMAIN SCACHE, supplies several configuration recommendations, a list of root name servers, and the Internet site from which the most current list can be obtained.

A FORWARDERS statement has also been added to improve name resolution capability for z/VM hosts that are connected to other hosts that provide network firewall services. This statement can be used to identify other name server hosts that can perform name resolution outside the firewall system.

In addition, the algorithms used in the caching subsystem have been improved to facilitate faster access to cached information and to more quickly determine when cached information does not exist.

SMTP Server

Changes Introduced in TCP/IP Level 3A0

- Timing values for the **RETRYAGE** and **WARNINGAGE** statements can now be specified in hours or minutes (in addition to a number of days), through the use of additional **H** or **M** statement operands. Existing defaults (specified in days, for which a **D** statement operand can now be specified) remain unchanged.

Changes Introduced in TCP/IP Function Level 320

- Support for the SMTP **EHLO** command has been added, as has support for the Message Size Declaration and 8–bit MIME transport service extensions. With this support, SMTP will now accept and handle SIZE and BODY parameters on MAIL FROM: commands.

Changes Introduced in TCP/IP Function Level 310

- The **DEBUG** configuration file statement is no longer supported. A new **TRACE** statement has been added which, when specified with the DEBUG parameter, will provide identical function as the DEBUG statement, except that output goes to only the console; there is no debug file support. Refer to “Chapter 20. Configuring the SMTP Server” on page 443 for more information on the **TRACE** statement and additional parameters that can be specified.
- The default on the **WARNINGAGE** configuration file statement has been reduced from 3 days to 1 day.
- Due to the introduction of new and changed trace options, the **TRACE** SMSG command now requires additional options to be specified. In TCP/IP releases prior to TCP/IP Function Level 310, the **TRACE** SMSG command was used to trace host name resolution via servers. This SMSG command has been replaced with the **TRACE RESOLVER** SMSG command. Similar results can be achieved by specifying the **TRACE RESOLVER** statement in the SMTP configuration file. Refer to “Chapter 20. Configuring the SMTP Server” on page 443 for more information about configuration file changes, and refer to the *TCP/IP User’s Guide* for SMTP SMSG command interface changes.
- The SMTP DATA file in no longer needed. The **ATSIGN** statement previously supported using this file has been incorporated within the TCPIP DATA file.
- The sample SMTP configuration file (SMTP SCONFIG) now specifies ″NETDATA″ as the default for the **LOCALFORMAT** and **RSCSFORMAT** statements, to reflect the operational default value associated with these statements.
- Host names for mail items can now be resolved (when required) by using the **REPROCESS** SMSG command. Refer to “Chapter 20. Configuring the SMTP Server” on page 443 for more information. The formerly documented SMTP-EXP EXEC is no longer needed for this purpose **and should not be used.**
- SMTP work files (NOTE and ADDRBLOK files) have new formats and names; the file types for these files have been changed to NOTEFILE and ADDRFILE, respectively. When migrating from IBM TCP/IP Version 2 Release 4, any NOTE or ADDRBLOK files present on the SMTP server A-disk will be converted to the new format and renamed; such converted files cannot be processed by previous versions of TCP/IP.

FTP Server

Changes Introduced in TCP/IP Level 3A0

- The FTP server now makes use of a server-specific configuration file (**SRVRFTP CONFIG** by default) for its startup information, in a manner more consistent with other TCP/IP servers. Therefore, `:Parms.` entries within the DTCPARMS file that are associated with the FTP server need to be modified to account for this change. Any SRVRFTP operands currently specified using a `:Parms.` entry need to be removed, with appropriate modifications made to either the FTP server configuration file or the DTCPARMS file (for example, to account for use of the ANONYMOU or RACF command operands).
- The FTP server has been enhanced to better accommodate web browser and graphical FTP clients. Changes for this support include the ability to provide

Unix-format list information in response to client DIR subcommand requests, through the use of the **LISTFORMAT** configuration statement. In addition, the **AUTOTRANS** statement can be used to configure the server to perform automatic EBCDIC-ASCII data translation for transferred files, based on file types (or, file extensions), as determined by **VMFILETYPE** statements, which are maintained in the TCPIP DATA file.

- Support for **AUTOTRANS** and **LISTFORMAT** operands to client-supplied SITE subcommands has been implemented also, to provide client control over automatic EBCDIC-ASCII data translation and list information formats.
- **AUTOTRANS** and **LISTFORMAT** characteristics can now be established when specific FTP users login, when the CHKIPADR exit exec is customized for use.
- Support for additional SMSG interface commands has been added to allow for dynamic server configuration changes, console and tracing control, and shutdown/restart capability.

Changes Introduced from TCP/IP Function Level 320

- The FTP server has been enhanced to exploit new CP and CMS user authorization facilities provided with VM/ESA Version 2 Release 4.0. These enhancements allow an FTP user to access minidisks they own without the need for minidisk passwords to be defined or supplied during an FTP session. Thus, the link results achieved when FTP users access minidisk resources may noticeably differ from those seen using prior levels of TCP/IP for VM.

  For example, if a user establishes a read-only link to a minidisk (through the use of an explicit “`ACCT` *`read_password`*”command), a subsequent PUT command that initiates a write request to that minidisk may now succeed *if* the login user ID owns the minidisk in question. By comparison, with a prior TCP/IP level, such a request would cause an FTP user to be prompted to supply a Read/Write (or Multiple Read) password through use of the **ACCT** subcommand in order to first gain write access to the minidisk.

  The use of these new user authorization facilities will also allow users with LOGONBY authority for a given *base* virtual machine to exercise that same authority during an FTP session. This is accomplished through use of the CP Diagnose X'88' command by the FTP server, which allows access to *base* user ID resources without requiring the password for that *base* user ID to be supplied. This kind of access is achieved through use of a new “`/BY/`*`byuser_id`*” parameter of the FTP **USER** subcommand. Additionally, the CP directory entry for the FTP server must include an `OPTION DIAG88` statement, to allow use of the Diagnose X'88' command.
- FTP virtual reader support has been added, which allows an FTP client to use the virtual reader of a VM user ID as the current directory. For such users to issue DELETE and DIR or LS commands against a reader directory, the FTP server virtual machine must have class D privileges. To allow users to PUT files to a reader directory, the **RDR** parameter must be included with the FTP server startup command, SRVRFTP.

FTP Client

Changes Introduced in TCP/IP Level 3A0

- The FTP client includes support for the **SIZE** command, which allows a client to obtain the size of a file before it is retrieved.

Changes Introduced in TCP/IP Function Level 320

- The FTP client has been enhanced to make use of login information present in a NETRC DATA file. By default, when a connection is made to a foreign host, user

ID and password information in the NETRC DATA file that is specific to that host will be used as part of an automatic FTP login process. Automatic FTP login can be suppressed using the new **NONETRC** option, or through the use of the new **NETRC** subcommand.

Printing

Changes Introduced in TCP/IP Function Level 310

- The LPR command can now route print files to RSCS for printing, freeing a user's virtual machine for other work. This may introduce the need for users to learn a limited set of RSCS commands in order to determine the status of their print files. Refer to the *TCP/IP User's Guide* for more information.
- RSCS can be used instead of LPSERVE to provide enhanced printer daemon (LPD) capabilities. The operation and administration characteristics of RSCS are very different from LPSERVE. Refer to "Chapter 15. Configuring the RSCS Print Server" on page 387 for more information.

Kerberos Server

Changes Introduced in TCP/IP Level 3A0

- The non-DES Kerberos functions that were provided with previous levels of IBM TCP/IP for VM are no longer available. Instead, only a DES Kerberos system is supported, with the DES Kerberos functions incorporated as part of the TCP/IP Feature for z/VM.

  Any Kerberos database that was created in a non-DES environment will not work with the DES Kerberos functions supplied as part of the TCP/IP Feature for z/VM. The existing non-DES Kerberos database **must** be deleted and then rebuilt using the supplied DES Kerberos functions. Refer to "Chapter 13. Configuring the Kerberos Server" on page 357 for more information about building the Kerberos database.

Implications of Assigning Different Server Virtual Machine Names

If any server *userid* is changed, the DTCPARMS file must be updated. Using the default server virtual machine names minimizes the amount of customization you must do prior to installation and when customizing the protocol server machine environments.

If site naming standards require names other than the default to be used, you can easily duplicate existing servers by using a DTCPARMS file and global exit that provides common server processing. You may choose to use the server profile exit facility, ensuring that server virtual machine names are changed. This requires modifying the customized configuration files and several product executables.

As you follow the configuration steps later in this publication, you must ensure the following changes:

1. The TCPIPUSERID statement in TCPIP DATA must be changed to reflect the actual user ID of the TCPIP server virtual machine.
2. A SYSTEM DTCPARMS file must be created that defines each server. Use the server definition section of the IBM DTCPARMS file as an example.
3. The SNALNKA GCS file for the SNALNKA virtual machine contains an assignment statement that must be changed if the default user ID TCPIP virtual machine name is changed. You must change the following line in the SNALNKA GCS file to the new user ID name:

```
TcpipUserid = 'TCPIP'
```

4. If RACF is active on the system on which TCP/IP is being installed, and you elect to use a different name for the File Transfer Protocol server (FTPSERVE) or the Network File System server (VMNFS), or you have multiple FTP or VMNFS servers installed, then the name (or names) must be accurately reflected in the FTPPERM EXEC or NFSPERM EXEC file. These files are located on the client code disk TCPMAINT 592. You must change the following line in the FTPPERM EXEC file to reflect the new names:

   ```
   ftpserve = 'FTPSERVE'
   ```

   You must change the following line in the NFSPERM EXEC file to reflect the new names:

   ```
   nfsserv = 'VMNFS'
   ```
5. The file VALIDUSR SEXEC, which is installed on the server common disk (TCPMAINT 591) contains processing code to determine if a particular user ID is authorized to exploit the SMSG interface to the Name Server. If used, the file should be copied to TCPMAINT 198 as VALIDUSR EXEC. As distributed, the exec assumes that the owner of TCP/IP is TCPMAINT. If you change the user ID of the owner to be something other than TCPMAINT, you will need to change the following line in the exec to reflect the new name:

   ```
   Call value thisnode'.TCPMAINT', !authuser
   ```
6. The file SNMPARMS NCCFLST, which is installed on the server code disk (TCPMAINT 591) contains initialization parameters for SNMP. If you customize the user ID of the virtual machine designated as the SNMP Query Engine, you must change the following line in the file to reflect the new SNMPQE name:

   ```
    SNMPQE  SNMPQE  * Userid of SNMP Query Engine
   ```

Publication References

Additional TCP/IP, z/VM, and VMSES/E product information can be obtained in the following publications:
- *z/VM: TCP/IP Level 3A0 Programmer's Reference*
- *z/VM: TCP/IP Level 3A0 User's Guide*
- *z/VM: TCP/IP Level 3A0 Messages and Codes*
- *z/VM: TCP/IP Level 3A0 Diagnosis Guide*

Chapter 2. System Requirements for TCP/IP

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

z/VM Device Definition Considerations

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

Hardware Environment

TCP/IP Feature for z/VM operates on any IBM S/390 or z/Architecture processor supported by z/VM (or any compatible processor). For information about supported processors, see *z/VM: General Information*.

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

Network Attachments

TCP/IP Feature for z/VM requires a network processor and associated components for attachments to the teleprocessing network. The following are possible network attachments that you may have installed or plan to install that work in conjunction with TCP/IP.

Open System Adapter 2 (OSA-2)

The IBM S/390 Open Systems Adapter (OSA) is an integrated hardware feature that allows the S/390 host platform to provide industry-standard connectivity to clients on directly-attached local area networks (LANs) or, via attachment to an asynchronous transfer mode (ATM) switch, to clients in an ATM-based network. The OSA-2 supports direct attachments to Fast (100Mb) Ethernet and IBM Token Ring LANs.

IBM 3172 Interconnect Controller

The IBM 3172 Interconnect Controller with the IBM Interconnect Controller Program Version 3 (5621-425), or any equivalent device that communicates with the host using the LAN Channel Station (LCS) protocol, is supported. One or more IBM Token Ring, Ethernet, or FDDI adapters must be installed.

Open System Adapter-Express (OSA-Express)

TCP/IP provides support for the OSA-Express, an integrated hardware feature providing fully integrated native systems connectivity to a Local Area Network (LAN) from a S/390 processor. Beginning with the IBM Generation 5 CMOS family of processors, a new type of I/O called the Queued Direct I/O Hardware Facility (QDIO) is available. QDIO allows the TCP/IP stack to directly exchange data with an I/O device without performing traditional S/390 I/O instructions. Data transfer is initiated and performed by referencing main storage directly via a set of data queues by both the I/O device and the TCP/IP stack. Once the TCP/IP stack establishes and activates the data queues, there is minimal processor intervention required to perform the direct exchange of data.

CLAW Channel Driver

Common Link Access to Workstations (CLAW) is a continuously executing, full-duplex channel program designed to minimize host interrupts while maximizing channel utilization. The channel commands used within CLAW allow S/390 applications, such as TCP/IP, to monitor the progress of a long channel program and modify it as it is running. In a S/390 processor with minimal CPU activity, CLAW is driven primarily by I/O interrupts, as typical S/390 devices would be. As the system load becomes heavier, CLAW is more often able to append new I/O requests to the already-active channel program — thus avoiding the overhead of interrupts and “Start Subchannel” requests at a time when it can least afford it. Therefore, CLAW provides better support of a direct hardware link between a S/390 channel and a workstation.

IBM 37xx Communications Controller Interface

The IBM 37xx Communications Controller Interface requires a 3720, 3725, or 3745 communication controller. The minimum IBM 37xx Communication Controller Software required to support the X.25 interface is:

- One of the following:
  - X.25 NPSI Version 3 (5688-035) Release 4 or later, for the 3745 or 3720
  - X.25 NPSI Version 2 (5688-719), for the 3725
- One of the following:
  - ACF/VTAM® for VM/ESA Version 4 (5654-010)
  - ACF/NCP Version 5 Release 3 or later (5668-738)
  - ACF/NCP Version 6 (5648-063).

HYPERchannel A220 Processor Adapter 42990007

TCP/IP supports the HYPERchannel Series A and B devices. In addition, TCP/IP supports HYPERchannel Series DX devices provided they function as Series A and B devices. For more information, see the appropriate Networking Systems Corporation documentation.

Channel-to-Channel Support

Channel-to-Channel connections are supported using the IBM 3088 Multi-system Channel Communication Unit. TCP/IP supports direct connection to another VM TCP/IP, MVS, OS/390, z/OS, or Linux for S/390 using the IBM 3088.

IUCV

The CP IUCV Service may be used to connect TCP/IP service machines on the same VM host. IUCV may also be used to connect VM TCP/IP to Linux for S/390.

Software Environment

TCP/IP Feature for z/VM requires:

- z/VM Version 3 Release 1.0 (TCP/IP Feature for z/VM cannot be used with previous levels of CP and CMS)

- Language Environment supplied with z/VM Version 3 Release 1.0 (previous levels of Language Environment cannot be used.)

To develop programs in Pascal or to modify TCP/IP Pascal components, use the IBM VS Pascal Version 1 Release 2, Compiler and Library (5668-767).

Note: The Pascal run-time library is included with TCP/IP. Therefore, it is not necessary to obtain the Pascal library just to use the TCP/IP feature.

To develop applications in the C programming language, or to modify TCP/IP C components, the following is required:
- IBM C forVM/ESA Compiler, Version 3 (5654-033) Release 1

Language Environment has been used to build the C components that provide the following TCP/IP services:
- Kerberos Servers
- Domain Name Server
- NDB Servers
- NFS Client and Servers
- PortMapper Server
- REXEC Daemon
- Sockets Applications Programming Interface
- SNMP Query Engine
- SNMP Agent
- RouteD Server
- Network Data Base Server
- MPROUTE Server

TCP/IP Feature for z/VM exploits a subset of RSCS Version 3 Release 2 function that is available with z/VM Version 3 Release 1.0. This subset provides enhanced printing support through RSCS LPR and LPD functions, along with TN3270E protocol support for printer sessions. This is the only intended or implied use of this subset of the RSCS product. Customers requiring the use of other RSCS functions must have or acquire an RSCS Version 3 Release 2 license.

If you are running a primary or secondary name server, you need DB2 Server for VSE & VM Version 6 (5648-A70) or DB2 Server for VSE & VM Version 7 (5697-F42). You do not need DB2 Server for VSE & VM if you are running a caching-only name server.

If the IBM Graphical Data Display Manager (GDDM) is to be run through the X Window System, the TCP/IP X Window System GDDM interface (GDDMXD) must be installed.

If you intend to install the CMS X Window System, you need:
- IBM C for VM/ESA Compiler Version 3 (5654-033)

The X Window System GDDM support requires one of the following:
- GDDM/VM Version 2.3 (5684-007) with APAR PN77393 applied
- GDDM/VM Version 3.1 (5684-168) with APAR PN77392 applied or later

Note: The X Window System server must be running Version 11.

System Requirements

For more information about GDDMXD, see “Chapter 28. X Window System Graphical Data Display Manager Support” on page 593.

If the IBM 3172 Support is to be used, the IBM 3172 Interconnect Controller Program Version 3 (5621-425) is required.

If the Network Data Base Server is to be used, the following is required:

- DB2 Server for VSE & VM Version 6 (5648-A70) or DB2 Server for VSE & VM Version 7 (5697-F42) running on the VM system.
- For the client workstation, AIX® Version 3 for RISC System/6000® (RS/6000®) or later, or a SunOS system for Sun workstations.
- For the server on VM, a CMS virtual machine with Class B privilege.

For SNAlink LU0 interface support, ACF/VTAM for VM/ESA Version 4 (5654-010) is required.

For X.25 interface support, the following is required:

- X.25 NPSI Version 3 (5688-035) Release 4 or later, for the 3745 or 3720
- X.25 NPSI Version 2 (5886-719), for the 3725
- Corresponding levels of ACF/VTAM and ACF/NCP that support NPSI

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

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

Chapter 3. General TCP/IP Server Configuration

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

Virtual Machine Definitions

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

Additional or differently-named virtual machines can be defined for your system to provide (or augment) the function provided by the "default" servers discussed in this section. For more information about defining such virtual machines and any applicable machine-specific requirements, see the *TCP/IP Feature for z/VM, Level 3A0 Program Directory*.

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

Table 1. Required TCP/IP Server Minidisk Links

Minidisk	Description
TCPMAINT 592	Client-code disk
TCPMAINT 591	Server-code disk
TCPMAINT 198	Configuration file, or *customization* disk

Note that much of the installation process for TCP/IP is completed as part of the installation of z/VM itself, so the "default" TCP/IP virtual machines described in this section, and the minidisk (or, DASD) resources they require, will have already been defined for your system. Thus, the directory definitions (supplied with z/VM) for the "default" TCP/IP virtual machines include links for the minidisks listed in Table 1.

Similarly, most commonly-used configuration files have been placed on the appropriate (production) minidisks and have been suitably named for use. However, these "production" configuration files must still be customized for your environment. These files are discussed further in "TCP/IP Configuration File Overview" on page 28.

Required Virtual Machines

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

Table 2. Required Virtual Machines

Machine	Function
3TCPIPA0	Maintains the TCP/IP system. Installation and service resources are owned by this user ID.

Table 2. Required Virtual Machines (continued)

Machine	Function
TCPIP	Provides TCP/IP communication services. The Telnet server is implemented as an ″internal client″ within the TCPIP virtual machine.
TCPMAINT	Owns TCP/IP production resources — the 198, 591, and 592 disks.

Optional Virtual Machines

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

Table 3. Optional Virtual Machines

Machine	Function	Page
ADMSERV	The Kerberos administration server.	357
BOOTPD	Responds to client requests for boot information contained in a BOOTP machine file.	255
DHCPD	Responds to client requests for boot information using information contained in a DHCP machine file.	271
FTPSERVE	Provides File Transfer Protocol (FTP) server support for controlled access to files on the local VM host.	339
LPSERVE	Provides Line Printer Daemon (LPD) support.	375
MPROUTE	The MPROUTE Server implements the OSPF protocol (OSPF Version 2) and/or the Routing Information Protocol (RIP) Version 2, and provides an alternative to static gateway definitions.	183
NAMESRV	The Domain Name server.	223
NDBPMGR	If an installation is going to run Network Database (NDB), you must install one NDBPMGR virtual machine and at least one NDBSRVn virtual machine, which is described in the next section. NDBPMGR is the NDB Port Manager virtual machine that coordinates the NDBSRVn virtual machines.	409
NDBSRVn	If an installation is going to run Network Database (NDB), you must install at least one NDBSRVn virtual machine and one NDBPMGR virtual machine, which is described in the previous section. NDBSRVn is the NDB server virtual machine that runs an SQL application by passing the database commands to SQL/DS.®	409
PORTMAP	Runs the Portmapper function for RPC on systems that support the Network File System protocol.	435

Table 3. Optional Virtual Machines (continued)

Machine	Function	Page
REXECD	The virtual machine for the REXECD server. It provides a remote execution service machine for TCP/IP hosts that support the REXEC client.	437
ROUTED	The RouteD Server implements the Routing Information Protocol (RIP) Version 1 and Version 2, and provides an alternative to static gateway definitions.	155
RXAGENT*n*	The agent virtual machines used by REXECD to execute commands from anonymous rexec clients.	437
SMTP	The virtual machine for the SMTP server. It receives mail over a TCP/IP network connection or from its virtual reader, and then sends that mail through the TCP/IP or RSCS network, according to the mail destinations.	443
SNALNKA	Provides SNA LU 0 connections between multiple hosts.	505
SNMPD	The SNMP Agent virtual machine.	513
SNMPQE	The SNMP Query Engine virtual machine.	513
SSLSERV	The virtual machine for the SSL server. (The SSL server runs on a specially-configured Linux operating system that runs in the SSLSERV virtual machine.)	525
TFTPD	Transfers files between the BFS (Byte File System) and TFTP clients. TFTPD supports access to files maintained in a BFS directory structure that is mounted during initialization.	547
UFTD	Implements the Unsolicited File Transfer (UFT) server.	565
VMKERB	The Kerberos Authentication server.	357
VMNFS	Implements the Network File System (NFS) server.	413
X25IPI	Runs a VTAM® application, Communication and Transmission Control Program (CTCP), which communicates to X.25 Network Control Program Packet Switching Interface (NPSI) through the DATE interface. This application allows internetworking between TCP/IP hosts using the X.25 protocol through the IBM 37xx communication controller.	583

Methods of Server Configuration

This section describes server configuration methods that allow you to add servers and server classes or modify the characteristics of a specific TCP/IP server virtual machine. Duplication of existing servers is also described. Configuration changes such as these are generally accomplished using the DTCPARMS file, described in

"The DTCPARMS File". Note that the DTCPARMS file is used only to configure CMS servers; GCS servers are configured using a *userid* GCS file, as explained in "GCS Servers" on page 28.

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

Protocol- or application-specific configuration support is provided through one or more server-specific configuration files. These files are listed in summary form in "TCP/IP Configuration File Overview" on page 28.

The DTCPARMS File

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

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

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

Table 4. DTCPARMS File Search

File	Purpose
userid DTCPARMS	May be used for servers that do not require configuration by the TCP/IP administrator, such as a test server. This file will most likely be found on a server's file mode A.
nodeid DTCPARMS	This is useful for shared-DASD configurations. The node ID used is the node ID returned by the CMS IDENTIFY command. This file should be kept on the TCPMAINT 198 disk.
SYSTEM DTCPARMS	Most server configurations should be kept in this file on the TCPMAINT 198 disk.
IBM DTCPARMS	Server classes provided by IBM, as well as the default server configurations, are in this file. This file is on TCPMAINT 591 and should never be modified because it will be replaced when service is applied or when a new release is installed. Any modifications you make should be placed in SYSTEM DTCPARMS.

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

In addition to the DTCPARMS files, server configuration information can be provided by server *profile exits*. See "Server Profile Exits" on page 27 for information.

Configuring the DTCPARMS File

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

DTCPARMS File Format

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

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

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

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

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

- Entries consist of *tags* and *tag values*.
- Entries that define a server using a `:Type.Server` definition must also include a `:Class.` tag and value to identify the *class* to which that server belongs.
- Tags defined as part of a *server* entry will be used for only that server instance (that is, the specific virtual machine user ID identified by the `:Nick.` tag).
- Tags defined as part of a *class* entry will be used for all servers of that class (unless overriding tags are defined as part of a *server* entry that references the class).
- | If multiple tags have the same name, the last one is used.
- Uppercase and lowercase characters can be used interchangeably for tags and most of their values.
- All tags do not apply to all servers.
- Any tag not recognized is ignored without warning.
- | Every entry must have a `:Type.tag.`
- Values cannot start with a colon, end with a period, or otherwise have the syntax of a tag.

DTCPARMS Tags

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

Table 5. DTCPARMS Tags for Configuring Servers

Tag	Description
:Anonymous.YES :Anonymous.NO	A value of **YES** will enable access to the server without requiring a VM user ID and password. The default is **NO**. Applies to the **nfs, ftp, rexec**, and **ndb** classes only.

Table 5. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:Attach.*raddr, raddr AS vaddr, raddr1-raddr2, ...*	A list of real addresses to be attached to this server. This server must have IBM privilege class A. Devices may be specified individually or as a range. Multiple devices or ranges must be separated by commas. Virtual address may be specified for a real address (not an address range) by appending "AS" followed by the virtual address. Example: **:attach.1500 AS 500, 400-403, 800-803** Applies to the **stack** class only.
:Class.*server_class*	Identifies the TCP/IP application protocol used by this server. Select from IBM-provided values or specify a protocol that has been defined by a **:type.class** entry. IBM-provided values: **stack, rip, dns, print, smtp, portmapper, rexec, rexec_agent, nfs, ftp, snmp, snmp_agent, snmpqe, ndb, ndb_agent, kerberos, kadmin, bootp, tftp, dhcp, uft.**
:Command.*command*	The command to run for this server.
:CSLibs.*csllib-list*	CSL libraries to be added to those implicitly defined by :runtime.
:DB2_Database.*dbname*	Identifies the DB2® database used by this server. SQLINIT will be issued for this database as well as a SET LANGUAGE (ADD ARI USER. Should be accompanied by the :vmlink. tag to access the DB2 run disk. For the **dns** class, do not specify a value for this tag if the domain name server is running in caching-only mode, which is the default.
:Diskwarn.*nn%* :Diskwarn.*nn%-fm* :Diskwarn.YES :Diskwarn.YES-*fm* :Diskwarn.NO	A warning is issued if less than **nn%** of read/write space is available on the indicated file mode. A value of **YES** only verifies that the file mode is accessed read/write. The percent sign is optional. The default is **NO**. If no file mode is specified, file mode A is the default.
:ESM_Enable.YES :ESM_Enable.NO	The External Security Manager (ESM) will be used to authenticate and authorize access to resources managed by this server. The default is **NO**.
:ESM_Racroute.YES :ESM_Racroute.NO :ESM_Racroute.*command*	The command to be used for initialization and termination of a RACROUTE environment. A value of **YES** causes the command **RPIUCMS** to be used. A value of **NO** indicates that no initialization of the RACROUTE environment is to be performed. The default is **NO**.
:ESM_Validate.YES :ESM_Validate.NO :ESM_Validate.*filename*	The name of a module to be used to validate user IDs and passwords supplied by clients. This module is preloaded into memory for improved performance. A value of **YES** causes **RPIVAL MODULE** to be used. A value of **NO** prevents loading of any module - RPIVAL module will be read from disk if **:ESM_Enable.YES** is specified. The default is **NO**.

Table 5. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:Exit.*exec-name*	The name of an exec to be run according to the rules defined in "Server Profile Exits" on page 27. The output from this exec overrides any specification returned by the global exit, TCPRUNXT EXEC.
:For.*userid*	The VM user ID of the rexecd server on whose behalf this server will do work. There is no default. Applies to the **rexecd_agent** class only.
:Loadlibs.*loadlib-list*	CMS LOADLIBs to be added to those implicitly defined by :runtime.
:Memory.nnnnK :Memory.nnnnM	The minimum virtual storage size required to run server. If the virtual machine is not large enough, an attempt will be made to define more storage and re-IPL CMS. If not specified, no check is performed.
:Mount.*bfs-pathname*	The fully-qualified Byte File System directory to be mounted as the root directory. Applies to the **tftp** class only. Values defined for this tag can be case sensitive.
:Name.*description*	Provides a description of the server to be displayed when the server is started. Mix cased values for this tag can be preserved.
:Nick.*userid* :Nick.*class_name*	For :**type.server**, the VM user ID for this TCP/IP server. For :**type.class**, an arbitrary name to be referenced by a :class. tag.
:Owner.*userid*	User ID to receive the console log. If an invalid user ID is specified, the log will be kept in the server's virtual reader. The default is **TCPMAINT**.
:Parms.*parameters*	Defines startup parameters to be passed to the server. Parameters should be specified as defined by the syntax of the command associated with this server. Parameters that affect the security characteristics of a server are automatically generated through the use of the **:Anonymous.** and **:ESM_Enable.** tags. Therefore, these parameters should not be specified using the `:Parms.` tag. See "Automatic Generation of Selected Startup Parameters" on page 25 for more information about parameters to which this applies. Parameters provided through the use of the `:Parms.` tag may override those that are generated automatically.
:Runtime.C :Runtime.PASCAL	**C** establishes the Language Environment for VM and MVS (SCEERUN LOADLIB). **PASCAL** establishes the VS PASCAL Release 2 runtime environment (TCPRTLIB LOADLIB). If this tag is not specified, a specific runtime language environment is not established.

Table 5. DTCPARMS Tags for Configuring Servers (continued)

Tag	Description
:Stack.*userid*	If specified, the user ID is compared to that given in the TCPIP DATA file. If it does not match, the server will not start.
:Type.class :Type.server	**Class** identifies this entry as an application protocol definition. **Server** identifies this entry as a server definition.
:vctc.*vaddr1* [TO] *userid vaddr2, ...*	Defines a virtual channel-to-channel device and couples it to the indicated virtual machine and address. The **TO** operand is optional. Example: **:vctc.200 to tcpip2 300, 201 tcpip2 301** Applies to the **stack** class only.
:VMLink.*vmlink-specification*	VMLINK-format nicknames, minidisks, or SFS directories to be accessed. See the *z/VM: CMS Command Reference* for information about VMLINK. Example: **:vmlink.* 195 tcpmaint 298 <298 G> (nonames**

Customizing Servers

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

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

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

- The following entry identifies the SQL1 DB2 database to be used by the NAMESRV server.

```
:Nick.NAMESRV   :Type.server  :Class.dns
           :DB2_Database.SQL1
           :VMlink.DB2DISK
           :Parms.MYDNB DATA M
```

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

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

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

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

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

```
:Nick.FTPSERVE  :Type.server  :Class.ftp
          :Diskwarn.NO
          :ESM_Validate.RPIVAL
          :ESM_Racroute.RPIUCMS
          :Name.FTP Server
```

Automatic Generation of Selected Startup Parameters

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

The server classes, default server IDs, startup parameters, and tags/values that affect this processing are listed in Table 6. For the servers listed in this table, the parameters indicated should be omitted from any `:Parms.` tag definitions used for those servers; the tags and values shown should be used instead, to allow these parameters to be generated during server initialization.

Note: Failure to use the tags listed in Table 6 may result in incorrect or insecure operation of the identified servers.

Table 6. Server Parameters Generated at Initialization

Server Class	Default Server ID	Generated Parameter	Controlling DTCPARMS Tag/Value
`rexec`	REXECD	-r -s	:ESM_Enable.YES :Anonymous.YES[1]
`nfs`	VMNFS	R N	:ESM_Enable.YES :Anonymous.YES
`ftp`	FTPSERVE	RACF ANONYMOU	:ESM_Enable.YES :Anonymous.YES
`ndb_agent`	NDBSRV*nn*	-r	:ESM_Enable.YES

Note[1]: For the -s parameter to be generated as a startup parameter for an REXEC server, the following conditions must be met:

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

Adding New Servers and Server Classes

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

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

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

Duplicating and Running Existing Servers

TCP/IP allows you to run more than one server of a given class, in the event such a need arises. Use the following steps to duplicate an existing server:

Server Duplication Steps

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

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

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

If necessary, consult the *TCP/IP Feature for z/VM, Level 3A0 Program Directory* for specific DASD storage and user ID requirements that may be applicable to the virtual machine in question.

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

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

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

```
AUTOLOG
  FTPSERVE  0
  FTPSERV2  0
  FTPSERVX  0

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

Note that existing entries for the FTPSERVE server are maintained.

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

Server Profile Exits

Occasionally you may find that you need more customization than can be provided by the DTCPARMS file. In that case you can use a *server profile exit*. This exit is a REXX exec that receives information about the server and returns configuration information. The name of the exit is specified on the `:Exit.` tag in DTCPARMS file.

Input

Information passed to the exit can be retrieved by the Arg(1) function or the PARSE ARG statement.

SETUP *class*
: The server class has been identified, but no commands have been issued. The console has been spooled to TCPMAINT. Use this exit point to access additional minidisks needed by the server or return any overriding DTCPARMS values.

BEGIN *class program*
: Called after the runtime environment has been established, immediately before the server program is started.

END *class return_code program*
: The server program has ended with the indicated return code.

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

ERROR *return_code command*
: The server cannot be started due to the failure of a needed CP or CMS command. The failing command, along with its return code is given. An error message has been displayed on the server console.

Output

The exit must return a return code and, optionally, values for tags in DTCPARMS. Any returned value overrides that specified in the DTCPARMS file. Values returned for END, ADMIN, or ERROR processing are ignored.

Return codes must be specified as the first token of a RETURN statement or on the EXIT statement. If no return code is given, it is assumed to be zero.

The following return codes are recognized:

0
: Server startup is to continue. For SETUP processing, any DTCPARMS value can be changed. For BEGIN, only changes to the `:Command.`, `:Parms.`, and `:Exit.` tags from DTCPARMS will be recognized.

4
: Server startup is cancelled and the server virtual machine will remain logged on.

For all other return codes, server startup is cancelled, and the server virtual machine will be logged off if running disconnected.

Examples

The following example does not specify a return code as the first token of the RETURN statement; therefore it is assumed to be 0 and server startup will continue:

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

The following example specifies a return code as the first token of the RETURN statement; therefore server startup will continue:

```
RETURN "0:Diskwarn.70%"
```

Global Profile Exit

If you have processing that is common to all servers, you can use the global profile exit, TCPRUNXT EXEC. It has the same inputs and outputs as the server profile exits. You can use the REXX `userid()` function to determine which server is starting, or use the class name argument that is passed to the exit. TCPRUNXT EXEC should be placed on TCPMAINT 198. TCPRUNXT SEXEC is provided as a sample. TCPRUNXT EXEC will automatically be invoked if it exists; therefore, it does not have to be identified within a DTCPARMS file.

The global exit can be used with or without individual server profile exits. If both are being used, the global exit is called before the server exit.

GCS Servers

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

TCP/IP Configuration File Overview

This section presents a table that lists the various TCP/IP configuration files that may be referenced by either TCP/IP clients, a specific TCP/IP server, or both. For each configuration file, Table 7 lists each (production) configuration file, its sample counterpart (if one is supplied by IBM), the minidisk where the *sample* file resides, and a reference to the chapter that provides details about the content and use of that file.

The first five files listed in Table 7 are necessary to provide basic TCP/IP services for most environments. The first file, IBM DTCPARMS, contains server configuration definitions. The next two files, PROFILE TCPIP and HOSTS LOCAL, | are configuration files for the TCPIP server virtual machine. The next two files, | TCPIP DATA and ETC SERVICES, need to be accessible to all TCP/IP servers, | applications, and users; these files contain information that is (or may be) | referenced by all users. ETC GATEWAYS contains routing information for distant networks and hosts. The remaining files are for various server virtual machines that you might have installed. Most sites will need to create the PROFILE TCPIP, HOSTS LOCAL and TCPIP DATA configuration files.

Table 7. Configuration Files and Minidisk Location Summary

Production Configuration File	IBM-Supplied Sample File	Disk	Reference Location
IBM DTCPARMS	none	591	"Chapter 3. General TCP/IP Server Configuration" on page 17
PROFILE TCPIP	PROFILE STCPIP	591	"Chapter 4. Configuring the TCP/IP Server" on page 33
HOSTS LOCAL	HOSTS SLOCAL	592	"Chapter 6. Configuring the HOSTS LOCAL File" on page 151

Table 7. Configuration Files and Minidisk Location Summary (continued)

Production Configuration File	IBM-Supplied Sample File	Disk	Reference Location
TCPIP DATA	TCPIP SDATA	592	"Chapter 5. Defining the TCP/IP System Parameters" on page 139
ETC SERVICES	ETC SAMPSERV	592	None; refer to comments within the supplied file.
ROUTED CONFIG	ROUTED SCONFIG	591	"Chapter 7. Configuring the ROUTED Server" on page 155
ETC GATEWAYS	none	-	"Chapter 7. Configuring the ROUTED Server" on page 155
NSMAIN DATA	NSMAIN SDATA	591	"Chapter 9. Configuring the DNS Server" on page 223
MASTER DATA	none	-	"Chapter 9. Configuring the DNS Server" on page 223
SRVRFTP CONFIG	SRVRFTP SCONFIG	591	"Chapter 12. Configuring the FTP Server" on page 339
LPD CONFIG	LPD SCONFIG	591	"Chapter 14. Configuring the LPD Server" on page 375
RSCSTCP CONFIG	RSCSTCP SCONFIG	591	"Chapter 15. Configuring the RSCS Print Server" on page 387
RSCSLPD CONFIG	RSCSLPD SCONFIG	591	"Chapter 15. Configuring the RSCS Print Server" on page 387
RSCSLPR CONFIG	RSCSLPR SCONFIG	591	"Chapter 15. Configuring the RSCS Print Server" on page 387
RSCSLPRP CONFIG	RSCSLPRP SCONFIG	591	"Chapter 15. Configuring the RSCS Print Server" on page 387
SMTP CONFIG	SMTP SCONFIG	591	"Chapter 20. Configuring the SMTP Server" on page 443
SMTP RULES	none	-	"Chapter 20. Configuring the SMTP Server" on page 443
UFTD CONFIG	UFTD SCONFIG	591	"Chapter 25. Configuring the UFT Server" on page 565
VMNFS CONFIG	VMNFS SCONFIG	591	"Chapter 17. Configuring the NFS Server" on page 413
PW SRC	none	-	"Chapter 22. Configuring the SNMP Servers" on page 513
SNMPTRAP DEST	none	-	"Chapter 22. Configuring the SNMP Servers" on page 513
X25IPI CONFIG	X25IPI SCONFIG	591	"Chapter 27. Configuring the X.25 Interface" on page 583

As mentioned previously, most commonly-used configuration files are copied to the appropriate (production) minidisks, and are suitably named for use when z/VM is installed.

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

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

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

Notes:

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

Starting and Stopping TCP/IP Servers

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

- TCP/IP stack, via the NETSTAT and OBEYFILE commands
- RouteD server, via an SMSG command interface
- MPROUTE server, via an SMSG command interface
- DNS server, via an SMSG command interface
- BOOTP server via console commands
- FTP server, via an SMSG command interface
- LPD server, via an SMSG command interface
- NFS server, via an SMSG command interface
- SMTP server, via an SMSG command interface
- SSL server, via its associated SSLADMIN command interface
- UFT server, via console commands.

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

Stopping TCP/IP Servers

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

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

   If you receive the message:

   ```
   DMSHDE744R Unexpected external interrupt detected, interrupt status consists of:
   CODE=0040, CPUID=0000, PARAMETER=00000000.  Enter a 1 for ABEND or a 2 for RESUME:
   ```

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

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

Note: The `#CP EXTERNAL` command should **not** be used to stop the RouteD or the SSL servers. The command to stop RouteD is `#CP SMSG * SHUTDOWN`. For more information on stopping the RouteD server, see "Using the SMSG Interface to RouteD" on page 179. The command to stop the SSL server is `SSLADMIN STOP`. For more information, see "SSLADMIN STOP Command" on page 544.

Starting TCP/IP Servers

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

Chapter 4. Configuring the TCP/IP Server

The TCPIP virtual machine provides the primary TCP/IP service called the *stack*. The stack supports the application programming interfaces and controls the network interfaces.

This chapter describes the statements and commands used to configure the TCP/IP stack. It also explains how you can dynamically change the configuration using the OBEYFILE command, as well as how to start and stop the TCP/IP stack.

To enable basic TCP/IP services, you must perform three activities.

Configuration Steps:

1. Configure the TCPIP virtual machine.
2. Define client system parameters in the TCPIP DATA file.
3. Create a site host table.

Client system parameters are explained in "Chapter 5. Defining the TCP/IP System Parameters" on page 139. The site host table is described in "Chapter 6. Configuring the HOSTS LOCAL File" on page 151.

Note: TCP/IP is an optional feature of z/VM Version 3 Release 1.0 that must be enabled to use it. For information on enabling this feature, see the *TCP/IP Feature for z/VM, Level 3A0 Program Directory*.

TCPIP Virtual Machine Configuration Process

To configure the TCPIP virtual machine, perform the following activities:

TCPIP Virtual Machine Configuration Steps:

1. Update the DTCPARMS file.
2. Create an initial configuration file.

Step 1: Update the DTCPARMS File

Change the DTCPARMS file to modify the run-time environment for the TCP/IP stack. You do not have to alter this file unless you want to change either the standard search order for the TCPIP server or the user ID of the virtual machine that receives its console log. See "Configuring the DTCPARMS File" on page 20 for more information about updating and configuring the DTCPARMS file.

Step 2: Create an Initial Configuration File

When the TCPIP virtual machine is started, TCP/IP operation and configuration parameters are read from an initial configuration file. TCP/IP first searches for file `node_name TCPIP`, where `node_name` is the system node name returned by the CMS IDENTIFY command. If this file is not found, TCP/IP uses file `PROFILE TCPIP` instead.

To customize your system, specify system operation, Telnet, and network parameters in this file, using the configuration statements listed in Table 9 on page 46. Complete statement syntax and descriptions are in alphabetical order in “TCP/IP Configuration Statements” on page 45.

You can also put many of these statements in a separate file and process it with the OBEYFILE command to dynamically change the TCP/IP configuration. For more information about OBEYFILE, see “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135. A sample initial configuration file is provided as `PROFILE STCPIP`. You can copy it from the server code disk, TCPMAINT 591, to TCPMAINT 198 and rename it to `PROFILE TCPIP` or *node_name* `TCPIP`. Then you can modify it to fit your requirements.

Routing Statements

TCP/IP supports static and dynamic routing. You can define static routes for your IP host using the GATEWAY statement in PROFILE TCPIP. You can also use the BSDROUTINGPARMS statement to configure RouteD to implement the Routing Information Protocol (RIPv1 and RIPv2) for dynamic routing. The functions of the GATEWAY and BSDROUTINGPARMS statements in PROFILE TCPIP are different in these two environments. Keep the following general rules and recommendations in mind when configuring RouteD:

- Use the GATEWAY statement in PROFILE TCPIP to define any routes that will not be used for dynamic routing (*static* routes). Using static routing is **not** recommended, since it prevents routes from being updated dynamically in response to network topology changes.
- If adjacent routers are not running RIP, define passive routes for them in the RouteD `ETC GATEWAYS` file so that RouteD does not modify them; these PASSIVE routes are then treated as static.
- Use the BSDROUTINGPARMS statement in PROFILE TCPIP to define the characteristics of each link used for dynamic routing.

MPROUTE is yet another way of providing dynamic routing implementing the OSPF v2 and RIPv1 and RIPv2 protocols. When configuring MPROUTE keep the following in mind:

- MPROUTE makes use of its own configuration file. Unlike RouteD, MPROUTE does not make use of the BsdRoutingParms, or GATEWAY configuration statements. The MTU, subnet mask, and destination address parameters are configured using the OSPF_Interface, RIP_Interface, and Interface statements within the MPROUTE configuration file.

Network vs. Host Routing: It is not recommended that you put multiple VM hosts with more than 1 hop count in the same subnet. This causes router confusion in selecting which path to choose to reach the destination. This confusion can be removed by using the -h parameter which enables RouteD to include host routes in the RIP updates. Another solution for this configuration is with static routes (GATEWAY statement) and not use RouteD. With the additional host route information in the RIP updates, the router (9.130.3.10) will use the host routes to reach the destinations, even though only one subnet route is being assigned to one of the interfaces.

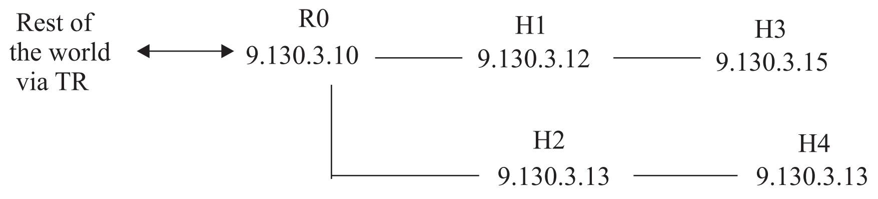


Figure 1. Host routing under single subnet.

The recommended practice for configurations involving destinations that are more than one hop count away from the source is to assign different subnets for each interface. For example,

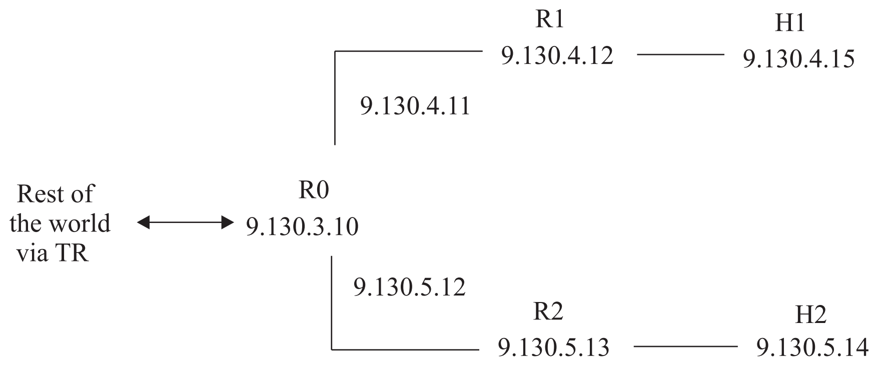


Figure 2. Subnet assignment for destinations beyond a single hop.

Because VM hosts (routers R1 and R2) are intermediate, they must be assigned different subnets (or networks). Notice that we have to assign new IP addresses 9.130.4.11 and 9.130.5.12 on R0 to define 9.130.4.x and 9.130.5.x subnets. This is necessary after expanding the network to reach other hosts. That is, assume that the original configuration is:

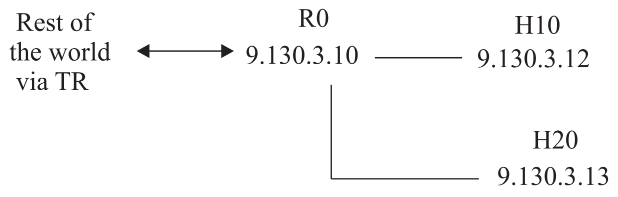


Figure 3. Basic host routing configuration.

This configuration is valid because VM hosts H10 and H20 are not routers, but hosts at end points from router R0. Now, when H1 and H2 are added to the configuration, H10 and H20 are no longer hosts by definition, but are routers. Therefore, in this case, you should assign different subnets to each routing interface as illustrated in Figure 2.

You can add more hosts in each subnet as long as those hosts do not become routers. For example, hosts H01, H10, and H21 are added to the configuration within their subnets:

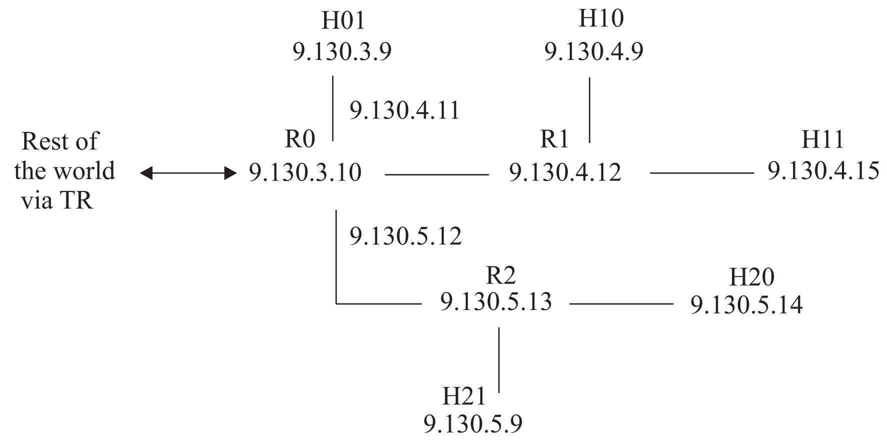


Figure 4. Adding hosts to subnetted interfaces.

A good reason for using different subnets for each routing interface is that routers communicate using network–specific routes, not host–specific routes. According to the RFCs for RIP, host—specific routing is optional. For this reason, there is no guarantee that routers in the network will communicate using host routes. That is, there are routers (for example, Wellfleet, Proteon) that will ignore host routes in the RIP broadcasts and rely on network–specific routes only. IP addresses, as well as their subnet masks, must be used with care when defining RIP networks. In addition, the -h parameter for advertising host routes in RouteD must be used carefully, and the network configuration must be planned with the assurance that the adjacent routers will accept the host routes.

Multicast Support

There are three methods that exist for sending data in a network environment:
1. Unicasting
2. Broadcasting
3. Multicasting

In unicasting, a datagram is sent to each host that requests it. A disadvantage of this is that the number of hosts is limited by the bandwidth on the sender. In broadcasting, copies of a message are sent to all hosts in the network, whether they request it or not. A disadvantage to this is that the hosts must be members of a single subnet.

In multicasting, a message is sent to multiple selected hosts in a group known as a multicast group. Multicasting provides the following advantages:
- Only one copy of a multicast message is sent over any link in the network; copies of a message are only made when paths diverge at a router.
- The sending server does not need to maintain a list of recipients because all hosts in a multicast group are identified by a single IP-destination address.
- Membership in a multicast group is dynamic:
  - Hosts can join or leave at any time
  - A host can be a member of more than one group at a time
  - The location of hosts within a network is unrestricted
  - The number of members in a multicast group is unlimited. When hosts join, you do not need to increase bandwidth.

- For applications that support IP multicast, a single group address can have multiple data streams on different port numbers on different sockets, in one or more applications. Multiple applications can share a single group address on a port.
- Multicasting reduces the load on the sending server, which no longer has to support multiple sequential or concurrent unicast sessions.

Note: TCP/IP contains host support of multicast, but not router support. Host support is based on implementation of the RFC 112 standard.

To implement multicasting in your applications:

- Use the C setsockopt() and getsockopt() calls to implement multicast functions. For information on these calls, refer to the *z/VM: TCP/IP Level 3A0 Programmer's Reference*.
- Use the GATEWAY statement in the PROFILE TCPIP configuration file to specify static routes for multicast datagrams. At a minimum, you should define a default multicast route.

Multiple Interface Network Support

TCP/IP supports multiple interfaces and IP addresses on the same network, providing redundant paths to other hosts or routers on directly-attached networks. When you configure multiple interfaces to the same network, one of them provides the primary path to hosts and routers on that network; the others are secondary paths.

In general, the interface used for traffic originating at your VM host is the one that provides the first active route to a destination, according to the IP routing table. If the destination route is not available, the interface that provides the first active default route is used. Furthermore, the home IP address of the selected interface is used as the local address for an application socket if SOURCEVIPA is not enabled and the socket is not bound to a specific local IP address.

If RouteD is in use and more than one directly-attached network interfaces is defined, the first link for a given home address is used as the primary source for routing outbound traffic to the local network, if that interface is functioning. All other interfaces on a directly-attached network are used for secondary routing in the event of an interface failure.

Notes:

1. The first primary path can be specified in the PRIMARYINTERFACE statement. On remaining interfaces, the first link for each defined network or subnetwork in the list of home addresses is the primary one.
2. If no PRIMARYINTERFACE statement is configured, the first link specified for each defined network or subnetwork in the list of home addresses is the primary one.

Virtual IP Addressing (VIPA)

Virtual IP Addressing (VIPA) frees other hosts from depending on a particular physical network interface for communication with a z/VM TCP/IP stack. Without VIPA, other hosts are bound to one of the VM host's home IP addresses and, therefore, to a particular physical network interface (for example, a device or adapter). If that interface fails, the associated connections are terminated. VIPA provides an IP address that is associated with a z/VM TCP/IP stack but not with a specific physical network interface. This allows hosts that connect to the z/VM

TCP/IP stack to send data on whatever paths are selected by the routing protocols; thus, VIPA provides tolerance of physical network interface hardware failures.

To achieve network interface independence, VIPA relies on a virtual device[1] and a virtual IP address. The virtual device is always active and never experiences a failure. A virtual IP address is the home address for a virtual device, which has no associated physical network interface. Inbound packets whose destination is a virtual IP address can be routed through any of the real physical network interfaces used by a z/VM TCP/IP stack. Failure of one physical network interface is handled by routing inbound traffic to another active physical network interface. Similarly, outbound packets can be routed around physical network interface outages, assuming an additional physical network interface provides an alternate path to the final destination.

Static or dynamic routing protocols may be used to manage alternate paths. In general, z/VM provides the following functions:

- Automatic and transparent recovery from device failure.

  When a device (for example, a 3172) fails, if there is another device that provides alternate paths to the destination, and if other hosts make connections using virtual IP addresses, then:

  – IP detects the failure, finds an alternate path for each network, and routes outbound traffic to hosts and routers on networks via alternate paths.
  – The result is fault tolerance for both inbound and outbound traffic, without the need to reestablish active connections that were using the failed device.
- Automatic and transparent recovery from adapter failure.

  Assume that multiple physical network adapters (e.g., Token Ring and FDDI) are installed on a device. If there are multiple alternate paths defined over these adapters to the destination, and if other hosts are connecting to virtual IP addresses, then:

  – IP detects the failure, finds an alternate path for each network, and routes outbound traffic to hosts and routers on those networks via alternate paths.
  – The result is fault tolerance for both inbound and outbound traffic, without the need to reestablish active connections that were using the failed adapter.
- Recovery from z/VM TCP/IP stack failure (when an alternate z/VM TCP/IP stack is configured to provide the necessary redundancy).

  Assume that an alternate TCP/IP stack is installed to serve as a backup and VIPA is configured on the primary TCP/IP stack. In case of a primary stack failure, the backup can be reconfigured to use the primary's virtual IP addresses. Client/server connections on the failed primary stack are disrupted but can be reestablished on the backup using the primary's virtual IP addresses as destinations. In addition, the temporarily reassigned virtual IP addresses can be restored to the original primary stack once recovery is complete.

  Note: For requests or connections originating at a z/VM TCP/IP stack, tolerance of device and adapter failures may be achieved by using the SOURCEVIPA feature. This capability causes virtual IP addresses to be used as the source IP addresses in all outbound datagrams except those associated with routing.

1. The term *virtual device*, used in this section to describe a device supported by VIPA, is not related in any way to VM's traditional virtual device support.

Figure 5 shows an example of a VIPA configuration.

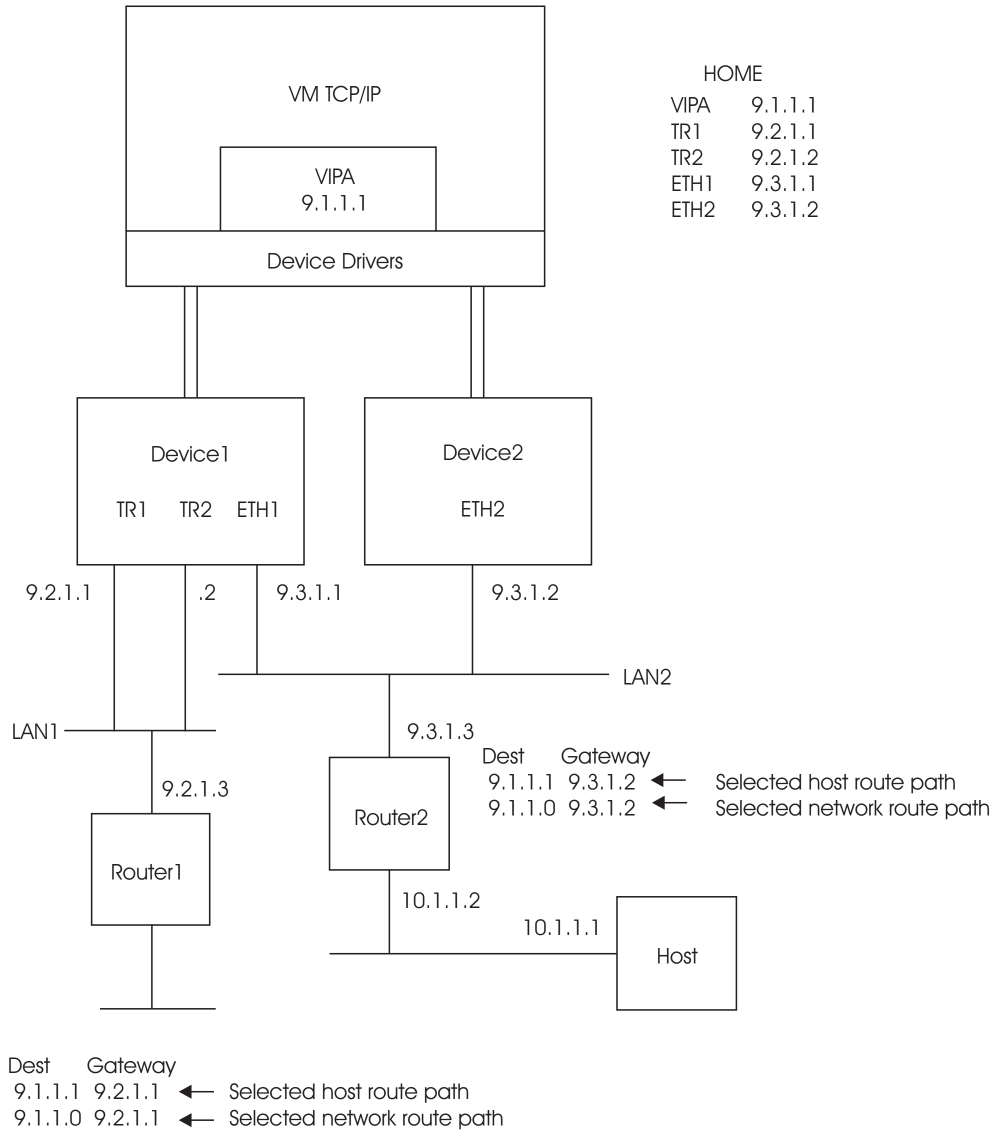


Figure 5. Single VIPA Configuration. Sample configuration showing multiple network interfaces using a single virtual IP address. Depending upon the routing protocols used, the network or host routes in Router1 and Router2 are used to reach the destination virtual IP address (9.1.1.1).

Configuring VIPA

Assume that you want to configure TCP/IP to use a virtual IP address. The necessary steps are:

1. Add a virtual device and link to the DEVICE and LINK statements. See "DEVICE and LINK Statement — Virtual Devices (VIPA)" on page 86.
2. Add the virtual link to the HOME statement. See "HOME Statement" on page 100.
3. If the virtual IP address is to be a default local host, see "PRIMARYINTERFACE Statement" on page 119.
4. A virtual link cannot be defined using the GATEWAY statement; therefore, there is no ″virtual″ route to define and no route to display via the NETSTAT GATE command.

5. If you want tolerance of device and adapter failures for requests or connections originating at a z/VM TCP/IP stack, specify the SOURCEVIPA option in the ASSORTEDPARMS statement. For this specification to be effective, the receiving nodes in the network must be configured to recognize the virtual IP addresses, using either static or dynamic routing protocols. Otherwise, timeouts for connection or request responses will occur as a result of the virtual IP addresses not being reachable. For more information on configuring home addresses using SOURCEVIPA, see “HOME Statement” on page 100.
6. For host name resolution, replace the physical IP addresses with the virtual IP address in the domain name servers.
7. Depending on the routing protocols in use, your routing paths— whether they are physical network interfaces used to reach destination hosts or those used by receiving nodes to reach the virtual IP address— must be either statically-defined or dynamically-learned and maintained. Consider the following:
   - If static routes are used (based on GATEWAY statement definitions), ensure that multiple paths are defined to reach a given destination host. Likewise, ensure that multiple paths are defined throughout your network to allow other hosts to reach the VIPA destination. The multiple paths are used to route around device and adapter failures. Refer to “GATEWAY Statement” on page 92 for additional information.
   - If RIP services (RouteD) are in use and Host Route Broadcasting (that is, the ability to learn host routes) is not supported by adjacent routers, the following restrictions for virtual IP addresses must be observed to gain the benefits of fault tolerance support:
     - If you use subnetting and virtual IP addresses are in the same network as the physical IP addresses, the subnetwork portions of virtual addresses must not be the same as the subnetwork portion of any physical IP addresses in the network. In this case, assign a new subnetwork for the virtual IP address.
     - If subnetting is not used on any physical interface, the network portion of any virtual IP addresses must not be the network portion of any physical IP addresses in the network. In this case, assign a new network for the virtual IP address, preferably a class C network address.
   - If RIP services (RouteD) are in use and Host Route Broadcasting is supported by adjacent routers, the network or subnetwork portions of virtual IP addresses can be the same across multiple z/VM TCP/IP stacks in the network. See “Chapter 7. Configuring the ROUTED Server” on page 155 for more information on how to configure Host Route Broadcasting on the RIP services.

Notes:
1. In addition to virtual links, physical links must be defined to provide the actual network attachments.
2. If static routing is used and multiple paths are defined through the network, the routers that provide the path information must have the ability to switch to alternate routes in the event of a failure. Otherwise, connections will be disrupted.

Using VIPA to Backup or Restore a TCP/IP Stack

Because a virtual IP address is associated with a TCP/IP stack and not with a specific physical network interface, a primary TCP/IP stack can be backed up by an alternate TCP/IP stack on the same or another z/VM system. This allows hosts that are connected to the primary TCP/IP stack to re-establish connections with an alternate stack when the primary is unavailable by using the virtual IP address of the primary. After the primary TCP/IP stack has been recovered, the temporarily reassigned virtual IP address can then be reclaimed from the alternate stack.

Whenever you back up or restore a z/VM TCP/IP stack, always consider the following:

- All sessions with servers on the failing host will be disrupted.
- Clients can use any ephemeral port number when connections are reestablished to backup servers.
- Having different port numbers for the alternate and primary servers is not recommended. If the alternate server has a different port number than the primary (for example, port 101 rather than port 21 for FTP), the client must know to use a different port (for example, 101 rather than 21). Using different port numbers does work, but can cause administrative problems.

For more information on backing up and restoring a z/VM TCP/IP stack, refer to the following application-dependent topics:

- If using static routing, see “GATEWAY Statement” on page 92.
- If using RouteD, see “Configuring a Backup TCP/IP Stack with VIPA” on page 179 and “Restoring a Primary TCP/IP Stack with VIPA” on page 179.

Native ATM Support

TCP/IP supports OSA-2 devices configured for HPDT ATM native mode operation. It supports both Private and Switched Virtual Circuits, as well as logical IP subnets. Unless the TRANSLATE statement is used to define the correspondence between ATM network IP and physical addresses, an ATMARP server must be available in each subnetwork to perform address resolution. TCP/IP for VM does not provide this function.

OSA Express Support

TCP/IP fully supports the OSA Express adapter which is available starting with the IBM Generation 5 CMOS processors. The OSA Express adapter provides integrated native systems connectivity to local Area Networks. TCP/IP provides two levels of support for the OSA Express Adapter:

1. Queued Direct I/O Hardware Facility (QDIO)
   - TCP/IP supports the following protocols for OSD Devices.
     - Gigabit Ethernet
     - Fast Ethernet
     - ATM
2. Legacy OSA LAN Emulation
   - TCP/IP support is provided through the traditional LCS device support.

Because the QDIO hardware facility allows for a direct connection between the TCP/IP stack and the OSA Express adapter resulting in improved performance, it is highly recommended that you configure the adapter as an OSD device.

DEVICE and LINK Statements

During the installation process, you must ensure that network devices are attached to the TCPIP virtual machine. You can accomplish this by either:

1. modifying the DTCPARMS file, enabling the necessary devices to be attached by using the `:Attach.` tag, or,
2. adding the appropriate DEDICATE control statements to the TCPIP virtual machine’s directory entry.

Note: A TCP/IP device address can be any hexadecimal value between 0001 and FFFF; a device address of 0000 is not valid.

TCP/IP for VM allows a single TCPIP virtual machine to drive multiple instances of a supported device. To configure your devices, add the appropriate DEVICE and LINK statements to the configuration file.

Unless otherwise noted, z/VM does not require a device definition in the system configuration file or HCPRIO. The actual device attributes are determined dynamically during device initialization. For more information on when and how devices are defined for z/VM, see *z/VM: Planning and Administration*.

The DEVICE statement identifies a device (control unit, communications adapter, or software service) that provides a connection to one or more networks or to a directly-connected host. Each connection to a particular network or host is called a *network interface*. The LINK statement identifies a specific network interface in a specific device, allowing TCP/IP to use it.

The HOME statement is used to assign a network address to the interface. The START and STOP statements are used to activate and deactivate the network interface. Each LINK statement should have corresponding HOME and START statements.

Devices are not automatically started when TCP/IP initializes, so you must either include START statements in the initial configuration file or start the devices manually using the OBEYFILE command.

There are DEVICE and LINK statements to configure the following:

Device Type	PageTopic
Asynchronous Transfer Mode (ATM) Network Connections	64
Channel-to-Channel Adapters (IBM 3088)	66
HYPERchannel A220 Devices	67
Integrated Communications Adapters	68
Local IUCV Connections	71
Remote IUCV Connections	72
LAN Channel Station (IBM 8232, 3172 or OSA)	73
Offload Host	77
| OSA Express (Gigabit Ethernet, Fast Ethernet, ATM)	80
RISC System/6000 IP Connections	82
SNA LU Type 0 Connections	85
Virtual IP Aaddressing (VIPA) Devices	86
IBM X.25 NPSI Connections	87

You can use DEVICE and LINK statements to connect two TCPIP virtual machines. See "DEVICE and LINK Statement — Local IUCV Connections" on page 71.

You can add new DEVICE and LINK statements using the OBEYFILE command but cannot modify any existing ones.

When you add new LINK statements, all the entries defined by the GATEWAY, HOME, BSDROUTINGPARMS, and TRANSLATE statements are deleted. Be sure to include the complete GATEWAY, HOME, BSDROUTINGPARMS, and TRANSLATE statements when adding new LINK statements with the OBEYFILE command.

For more information about OBEYFILE, see “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135.

Point-to-Point Connections to Other Hosts: A point-to-point connection is a network that consists of exactly two hosts. As with traditional networks, each host must assign a network address to its network interface. The address assigned does not need to be unique as long as the host has some other network connection for which a unique address has been assigned and you are not running RouteD or MPROUTE. In this case, the point-to-point link can be considered an “extension” of another.

For example, consider the following scenario:

- Hosts A and B are connected by SNA LU Type 0
- Host A is also connected to a token-ring whose address is 193.1.1
- Host B is also connected to a token-ring whose address is 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 6. Point-to-Point Link

Host A’s configuration file could contain:

```
home
    193.1.1.1   tr1
    193.1.1.1   snalu0

gateway
;  Network     First hop   Link     Packet Size     Subnet mask
   193.1.1         =       tr1         2000             0
   193.1.2         =       snalu0      2000             0
```

Host B’s configuration file could contain:

```
home
    193.1.2.1   tr1
    193.1.2.1   snalu0

gateway
;  Network     First hop   Link     Packet Size     Subnet mask
   193.1.2         =       tr1         2000             0
   193.1.1         =       snalu0      2000             0
```

Note that the SNA link does not have its own home address. Hosts A and B are addressed by their token-ring addresses, even if the packets reach them through the SNA link.

If host B had no other network attached to it you would have to assign a separate (sub)network number to the SNA link. Even in this case, Host A does not need a separate home address for its side of the link because it can be addressed by its token-ring home address. Host B's only home address is the home address for the SNA link.

Free Pool Statements

Each control block or data buffer needed by TCP/IP is allocated from the *free pool*, the size of which is determined by the virtual storage size of the TCPIP virtual machine. The free pool is subdivided into separate pools for each type of control block or buffer. The initial size of each pool is determined by a pool configuration statement. All pools are created when TCP/IP is started, so if there is not enough virtual storage to contain their initial sizes, TCP/IP will not start.

As shown in Table 8, most pools have an associated *permit size*, or threshold, which is computed as a percentage of the pool size. When the number of elements (control blocks or data buffers) remaining in a particular pool drops below the permit size, TCP/IP will send a message to every user in the INFORM list. The message is sent only once per pool until the NETSTAT RESETPOOL command is issued. Table 8 also shows the *limit size* associated with each pool. This value is either a percentage of the pool size or an absolute value and is smaller than the permit size. When the number of elements left in a pool drops to its limit size, TCP/IP attempts to allocate more pool elements dynamically and sends a message to every user in the INFORM list to report the situation and indicate whether it was alleviated.

The NETSTAT POOLSIZE command displays the total number of elements, the number of elements in use, the minimum number of elements available since TCP/IP was started, and the permit size of each pool. Use this command to monitor and adjust pool sizes to ensure availability of TCP/IP services.

Table 8. Free Pool Configuration Statements. Each pool configuration statement is shown. The inform threshold is the percentage used to calculate the permit size. The limit is either the percentage of the pool size used to calculate the limit size or an absolute number of pool elements.

Statement	Inform Threshold	Limit	Page
ACBPOOLSIZE	10%	5%	48
ADDRESSTRANSLATIONPOOLSIZE	0.33%	0%	49
CCBPOOLSIZE	10%	5%	62
DATABUFFERPOOLSIZE	10%	5%	63
ENVELOPEPOOLSIZE	10%	5%	89
FIXEDPAGESTORAGEPOOL	10%	0%	91
IPROUTEPOOLSIZE	2%	0%	108
LARGEENVELOPEPOOLSIZE	10%	5%	110
RCBPOOLSIZE	10%	5%	120
SCBPOOLSIZE	10%	5%	122
SKCBPOOLSIZE	10%	5%	123
SMALLDATABUFFERPOOLSIZE	10%	5%	124
TCBPOOLSIZE	10%	5%	127
TINYDATABUFFERPOOLSIZE	10%	5%	128
UCBPOOLSIZE	10%	5%	135

Routing Statements

TCP/IP supports static and dynamic routing. Static routes to other TCP/IP hosts are defined using the GATEWAY configuration statement. Dynamic routing is provided by the RouteD service, implementing the Routing Information Protocol (RIP). RouteD uses the BSDROUTINGPARMS configuration statement, not GATEWAY. The statements that affect routing are:

Statement	Page
ARPAGE	50
BSDROUTINGPARMS	59
GATEWAY	92
HOME	100
PRIMARYINTERFACE	119

See "Chapter 7. Configuring the ROUTED Server" on page 155 for details.

Tracing Statements

TCP/IP provides the capability to log various events that occur in the TCPIP virtual machine. Tracing should normally be turned off, but may be requested by the TCP/IP service group. The trace output can be directed to the TCPIP virtual machine console or to a disk file. The trace-related configuration statements are:

Statement	Page
FILE	90
LESSTRACE	111
MORETRACE	113
NOSCREEN	114
NOTRACE	114
SCREEN	123
TIMESTAMP	128
TRACE	130
TRACEONLY	131

TCP/IP Configuration Statements

This section describes the statements you use to customize the TCP/IP stack and reflect your installation's network configuration.

Configuration Statement Syntax

Statement syntax is the same in both the configuration file and an obey file. The following formatting restrictions apply to configuration statements:

- Statements are free format; leading blanks, comments, and end-of-record are ignored.
- A configuration statement consists of a statement name followed by a required blank and usually one or more positional arguments. Separate arguments with one or more blanks.
- A semicolon, followed by a blank, begins a comment. Comments act as blanks, separating words without affecting their meaning.
- Arguments followed by comments must have a blank before the semicolon.
- Statements can be split across multiple lines.
- Sequence numbers are not allowed.
- Lower-case letters are translated to upper case before a statement is processed.
- Abbreviations of statement names are not allowed.
- An END*statement* terminates several statements, such as AUTOLOG and ASSORTEDPARMS. If the END*statement* is omitted, all subsequent tokens in the file are interpreted as parameters of that configuration statement.

Summary of TCP/IP Configuration Statements

Table 9. Summary of TCP/IP Configuration Statements

Statement	Description	Page
ACBPOOLSIZE	Defines the initial number of activity control blocks in the free pool	48
ADDRESSTRANSLATIONPOOLSIZE	Defines the initial number of address translation control blocks in the free pool	49
ARPAGE	Defines the number of minutes before an ARP table entry is deleted	50
ASSORTEDPARMS	Defines miscellaneous TCP/IP parameters	50
ATMARPSERVER	Defines the location of an outboard ATMARP server	54
ATMLIS	Defines a logical IP subnet of an ATM network	55
ATMPVC	Defines a permanent virtual circuit for an ATM network	56
AUTOLOG	Supplies the names of additional virtual machines to be started when TCP/IP is initialized	57
| BLOCK	Specifies IP addresses from which traffic is to be blocked.	58
BSDROUTINGPARMS	Defines network interface information for the RouteD server	59
CCBPOOLSIZE	Defines the initial number of client control blocks in the free pool	62
DATABUFFERLIMITS	Specifies the maximum number of data buffers that may be allocated for a TCP connection that uses window scaling	63
DATABUFFERPOOLSIZE	Defines the initial number of data buffers in the free pool	63
DEVICE and LINK	Defines an interface to a network or host	
Asynchronous Transfer Mode (ATM) Network Connections		64
Channel-to-Channel Adapter		66
HYPERchannel A220 Devices		67
Integrated Communications Adapters		68
Local IUCV Connections		71
Remote IUCV Connections		72
LAN Channel Stations (IBM 8232, IBM 3172, or OSA		73
Offload Hosts		77
OSA Express		80
RISC System/6000 IP Connections		82
SNA LU Type 0 Connections		85
Virtual IP Addressing (VIPA)		86
IBM X.25 NPSI Connections		87

Table 9. Summary of TCP/IP Configuration Statements (continued)

Statement	Description	Page
ENVELOPEPOOLSIZE	Defines the initial number of envelopes in the free pool	89
FIXEDPAGESTORAGEPOOL	Defines the initial number of Fixed Page Storage Blocks as well as the maximum to be placed in the pool.	91
FILE	Specifies a file to receive trace information	90
GATEWAY	Indicates how to route datagrams to specified networks	92
HOME	Defines a list of home addresses and associated link names	100
INFORM	Lists users who are to be informed in case of serious run-time conditions	103
INTERNALCLIENTPARMS	Configures the Telnet server, an internal client of TCP/IP	105
IPROUTEPOOLSIZE	Defines the initial number of IP route control blocks in the free pool	108
KEEPALIVEOPTIONS	Specifies the operating parameters of the TCP keep-alive mechanism	109
LARGEENVELOPEPOOLSIZE	Defines the initial number of large envelopes in the free pool	110
LESSTRACE	Turns off detailed tracing of specified TCP/IP processes	111
MONITORRECORDS	Controls which monitor data records are generated	112
MORETRACE	Turns on detailed tracing of specified TCP/IP processes	113
NOSCREEN	Directs trace output to the current trace file on disk	114
NOTRACE	Turns off all tracing for specified TCP/IP processes	114
OBEY	Identifies users who can use privileged TCP/IP commands and services	115
PERMIT	Identifies users who can use TCP/IP services	116
PORT	Assigns a port to one or more servers	117
PORT Statement for Telnet	Assigns a port to the internal Telnet server	118
PRIMARYINTERFACE	Specifies which link is the primary interface	119
RCBPOOLSIZE	Defines the initial number of Raw IP control blocks in the free pool	120
RESTRICT	Lists users who are prohibited from using TCP/IP	121
SCBPOOLSIZE	Defines the initial number of socket control blocks in the free pool	122
SCREEN	Directs trace output to the console	123
SKCBPOOLSIZE	Defines the initial number of socket interface control blocks in the free pool	123
SMALLDATABUFFERPOOLSIZE	Defines the initial number of small data buffers in the free pool	124
START	Starts the specified device	125
STOP	Stops the specified device	125

Table 9. Summary of TCP/IP Configuration Statements (continued)

Statement	Description	Page
SYSCONTACT	Specifies the value of the MIB-II variable sysContact, which contains information about the TCP/IP administrator for this host	126
SYSLOCATION	Specifies the value of the MIB-II variable sysLocation, which contains information about the physical location of this host	126
TCBPOOLSIZE	Defines the initial number of TCP control blocks in the free pool	127
TIMESTAMP	Determines how often time stamps are displayed with messages	128
TINYDATABUFFERPOOLSIZE	Defines the initial number of tiny data buffers in the free pool	128
TN3270E	Defines client IP addresses and logical unit names that may establish printer sessions	129
TRACE	Identifies internal TCP/IP processes for run-time tracing	130
TRACEONLY	Restricts TCP/IP tracing to certain users, devices, or IP addresses	131
TRANSLATE	Indicates the relationship between an IP address and a network address	133
UCBPOOLSIZE	Defines the initial number of UDP control blocks in the free pool	135

ACBPOOLSIZE Statement

Use the ACBPOOLSIZE statement to set the initial number of activity control blocks (ACBs). ACBs are used to schedule processes within the TCPIP virtual machine. Each one requires 112 bytes.

```
   ┌─ACBPOOLSIZE 1000───┐
►►─┼────────────────────┼──────────────────────────────────────────►◄
   └─ACBPOOLSIZE number─┘
```

Operands

number
The initial number of ACBs in the free pool. The default is 1000. The minimum number is 100.

Examples

This example shows an ACBPOOLSIZE statement that defines the number of ACBs to be the default of 1000.

```
ACBpoolSize   1000
```

Usage Notes

- As long as you do not specify NOACBCUSHION on the ASSORTEDPARMS statement, the system will attempt to dynamically allocate 10% more ACBs any time the ACB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many ACBs your system is using. To avoid dynamic allocation of ACBs during operation, use the ACBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- "ASSORTEDPARMS Statement" on page 50
- *TCP/IP User's Guide* for the NETSTAT command

ADDRESSTRANSLATIONPOOLSIZE Statement

Use the ADDRESSTRANSLATIONPOOLSIZE statement to set the initial number of address translation control blocks. Address translation control blocks are used to hold information about the relationship between IP addresses and network addresses. Each one requires 153 bytes.

```
    ┌─ADDRESSTRANSLATIONPOOLSIZE 1500───┐
►►──┤                                   ├──────────────────────────────────►◄
    └─ADDRESSTRANSLATIONPOOLSIZE number─┘
```

Operands

number
The initial number of address translation control blocks in the free pool.

Examples

This example shows an ADDRESSTRANSLATIONPOOLSIZE statement that defines the number of address translation control blocks to be the default of 1500.

```
AddressTranslationPoolSize   1500
```

Usage Notes

- Each entry in the ARP table, whether entered using the TRANSLATE statement, created dynamically via ARP, or added by a device driver as a home address translation entry, requires one address translation control block.
- The system will attempt to dynamically allocate 10% more address translation control blocks any time the address translation control block free pool becomes empty. You can use the NETSTAT POOLSIZE command to monitor how many address translation control blocks your system is using. To avoid dynamic allocation of address translation control blocks during operation, use the ADDRESSTRANSLATIONPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- "TRANSLATE Statement" on page 133
- *TCP/IP User's Guide* for the NETSTAT command

ARPAGE Statement

Use the ARPAGE statement to determine how long an ARP table entry is retained after it is created or revalidated. By default, TCP/IP deletes ARP table entries five minutes after they are created or revalidated. An ARP table entry is revalidated when another ARP packet is received from the same host specifying the same hardware address.

```
      ┌─ARPAGE 5───────┐
►►────┼────────────────┼─────────────────────────────────────────────────►◄
      └─ARPAGE minutes─┘
```

Operands

minutes
The number of minutes between creation or revalidation of an ARP table entry and its deletion. The default is 5.

Examples

This example clears the ARP tables every 5 minutes.

```
arpAge 5
```

Usage Notes

This number is an integer from 1 to 99,999,999.

ASSORTEDPARMS Statement

Use the ASSORTEDPARMS statement to pass initialization parameters to TCP/IP. Any misspelled or otherwise undefined parameters are ignored and no error message is generated. Valid parameters are acknowledged by a message.

```
            ┌──────────────────────────┐
            ▼                          │
►►──ASSORTEDPARMS────┬──────────────────────┬──ENDASSORTEDPARMS───────────►◄
                     ├─CHECKCONSISTENCY─────┤
                     ├─CLAWUSEDOUBLENOP─────┤
                     ├─┬─CPDUMP─┬───────────┤
                     │ └─VMDUMP─┘           │
                     ├─IGNOREREDIRECT───────┤
                     ├─NOACBCUSHION─────────┤
                     ├─NOFWD────────────────┤
                     ├─NORFC1323────────────┤
                     ├─NOUDPQUEUELIMIT──────┤
                     ├─OVERRIDEPRECEDENCE───┤
                     ├─PERMITTEDUSERSONLY───┤
                     ├─PROXYARP─────────────┤
                     ├─RESTRICTLOWPORTS─────┤
                     ├─SOURCEVIPA───────────┤
                     ├─STOPONCLAWERROR──────┤
                     └─VARSUBNETTING────────┘
```

Operands

CHECKCONSISTENCY
Consistency checking performs periodic internal checks to ensure that data structures within the TCPIP virtual machine are intact. Making these checks necessarily involves referencing a potentially large number of pages and can cause undesirable intermittent bursts of paging activity. Specify this parameter if there is reason to believe that internal inconsistency is a source of other problems, in order to attempt to detect these earlier. The consistency check setting is confirmed with one of the messages:

```
Internal consistency checking is enabled
Internal consistency checking is disabled
```

CLAWUSEDOUBLENOP
Causes channel programs for Common Link Access to Workstation (CLAW) devices to end with two NOP CCWs instead of one. This is required for some vendor devices.

CPDUMP
Causes any dump created as a result of a program check to be generated using the CP DUMP command.

The CPDUMP parameter is confirmed by the message:

```
TCP/IP will take dump in case of program check
```

VMDUMP
Causes any dump created as a result of a program check to be generated using the CP VMDUMP command.

The VMDUMP parameter is confirmed by the message:

```
TCP/IP will take VMDUMP in case of program check
```

If neither VMDUMP nor CPDUMP is specified, no dump will be generated. This situation is confirmed with the message:

```
TCP/IP will not dump if a program check occurs
```

ASSORTEDPARMS

IGNOREREDIRECT
Causes TCP/IP to ignore ICMP Redirect packets. The IGNOREREDIRECT setting is confirmed by one of the messages:

```
ICMP will ignore redirects
ICMP will honor redirects
```

If you are using RouteD, use this option since RouteD does not support ICMP redirects.

NOACBCUSHION
Prevents dynamic expansion of the ACB pool, so that TCP/IP terminates if the pool is exhausted. Normally, when the number of free ACBs drops below 5%, the system attempts to allocate more of them. You should not specify this parameter in ordinary circumstances.

NOFWD
Stops the transfer of data between networks by disabling IP datagram forwarding. This statement can be used for security or to ensure correct use of limited resources.

If NOFWD is not specified, IP packets will be forwarded when this host is a gateway.

If you are using RouteD, it is recommended that you not use this option.

The IP forwarding setting is confirmed by one of the messages:

```
IP forwarding is enabled
IP forwarding is disabled
```

NORFC1323
Prevents the initiation of window scaling and associated features for high-performance TCP connections, as specified bu RFC 1323. If NORFC1323 is not specified, TCP/IP tries to enable window scaling and related TCP performance options for connections it initiates. Requests from other hosts to enable these facilities are always accepted.

The state of RFC 1323 support is reported by one of the messages:

```
Support for RFC 1323 is enabled
Support for RFC 1323 is disabled
```

NOUDPQUEUELIMIT
Causes TCP/IP to relax the limit of 21 incoming datagrams queued on a UDP port. If you specify NOUDPQUEUELIMIT when you are running untested applications on your system, a malfunctioning application can tie up the available envelopes (or the available data buffers if you are using Offload).

The state of the NOUDPQUEUELIMIT parameter is confirmed by one of the messages:

```
Limit on incoming UDP datagram queue size enabled
Limit on incoming UDP datagram queue size disabled
```

OVERRIDEPRECEDENCE
| For TCP connections initiated or accepted by VM TCP/IP, allows an external
| client to alter the precedence for a connection (when the connection is first
| established or at anytime throughout its duration), overriding any default
| value set by the VM host. This violates the TCP protocol, as specified by RFC 793, but is necessary in some environments.

If this option is not specified, the precedence designation in all incoming
| datagrams must match the precedence designated by the VM TCP/IP host
| when it initiates or accepts a connection.

The OVERRIDEPRECEDENCE setting is confirmed by one of the messages:

```
Precedence designations will be overridden
Precedence designations will be respected
```

PERMITTEDUSERSONLY

Restricts the use of TCP/IP services to only those users who are explicitly identified in the initial configuration file or any subsequent obey file.

Without this parameter, any user not specified on the RESTRICT statement may use TCP/IP services.

The state of the PERMITTEDUSERSONLY parameter is confirmed by one of the messages:

```
Only users mentioned in PROFILE TCPIP may use TCP/IP services
Access to TCP/IP services is not restricted
```

PROXYARP

Causes the z/VM host where TCP/IP is running to act as a proxy for guest hosts that use point-to-point connections (for example, Channel-to-Channel and IUCV connections). ARP requests for the MAC address associated with the IP address of a point-to-point connection are answered by TCP/IP on behalf of the guest, which causes traffic to be routed to VM for forwarding to the ultimate destination.

PROXYARP should be specified when the IP address for a host associated with a point-to-point connection is in the same subnet as other IP addresses that are in use on LANs to which the z/VM host (that is providing guest support) has a connection.

RESTRICTLOWPORTS

Restricts the use of available well-known port numbers (0 through 1023) to users who are specified on the OBEY statement who have a port explicitly reserved for them on a PORT statement. If this parameter is not specified, any unreserved port can be used by any user.

The state of the RESTRICTLOWPORTS parameter is confirmed by one of the messages:

```
Access to ports 0-1023 is not restricted
Only users in the obey list may use ports 0-1023
```

SOURCEVIPA

Requests TCP/IP to use the closest virtual IP address to the actual destination address in the HOME list as the source IP address for outbound datagrams. This parameter has no effect on RIP servers, such as RouteD, for routing protocol packets.

STOPONCLAWERROR

Terminates TCP/IP immediately when a CLAW device error is detected. If this parameter is specified, it is confirmed by the message:

```
TCP/IP will halt on certain CLAW errors
```

VARSUBNETTING

Enables variable subnetting and supernetting support when RouteD is being used, and allows variable-length subnet masks to be specified in GATEWAY and BSDROUTINGPARMS statements. When used, the ASSORTEDPARMS statement must be placed before the GATEWAY and BSDROUTINGPARMS statements. By default, variable subnetting is not supported.

Inclusion of the VARSUBNETTING parameter also allows variable-length masks to be included in RIP Version 2 packets through RouteD dynamic updates to the IP routing table. If RouteD is configured to use RIP Version 2, the VARSUBNETTING operand must be specified as part of the ASSORTEDPARMS statement.

Variable-length subnetting allows different subnets to use subnet masks of differing sizes. Thus, a subnet can make use of a mask that is appropriate to its size, in order to maximize the number of host addresses that can be allocated and used within that subnet (or conversely, to avoid subnets that use only a small portion of the host addresses available within that subnet).

Refer to “Configuring Multiple Subnets with the Same IP Address” on page 174 for an example of a variable subnetted local network.

Examples

This example shows two parameters on the ASSORTEDPARMS statement.

```
AssortedParms
  NOFWD
  RestrictLowPorts
EndAssortedParms
```

Usage Notes

- The ENDASSORTEDPARMS statement is the delimiter for the ASSORTEDPARMS statement. If ENDASSORTEDPARMS is omitted, subsequent statements are saved until the maximum parameter string length is reached, and then an error message is generated. The ASSORTEDPARMS string is then cleared, and processing resumes at the next recognized configuration statement.
- When using the OBEYFILE command to modify the ASSORTEDPARMS statement, keep these rules in mind:
  – The values of all parameters are changed.
  – An ASSORTEDPARMS parameter that is not specified assumes its default value or setting.

Context

- “PORT Statement” on page 117
- “INFORM Statement” on page 103
- “OBEY Statement” on page 115
- “RESTRICT Statement” on page 121
- “PERMIT Statement” on page 116
- “OBEYFILE Command” on page 135

ATMARPSERVER Statement

Use the ATMARPSERVER statement to define an outboard ATMARP server that can be used to resolve ATMARP requests for a logical IP subnetworkof an ATM network.

```
►►──ATMARPSERVER──server_name──subnet_name──────────────────────────────►

►──┬─PVC──pvc_name──────────────────────────────┬──────────────────────►◄
   └─SVC──ip_address──NSAP──physical_address────┘
```

Operands

server_name
The one- to 16-character name of this ATMARP server.

subnet_name
The name of the logical IP subnet, defined by an ATMLIS statement (see "ATMLIS Statement"), that this ATMARP server manages.

PVC *pvc_name*
The name of the private virtual circuit, defined by an ATMPVC statement (see "ATMPVC Statement" on page 56), to this ATMARP server.

SVC *ip_address*
The IP address, in dotted decimal form, of the ATMARP server to which a switched virtual circuit is to be established.

NSAP *physical_address*
The 40-hexadecimal-digit physical address of the ATMARP server.

Examples

This example shows an ATMARPSERVER statement defining a server that communicates via an SVC connection.

```
ATMArpServer MSS1 SUBNETC
             SVC  125.3.0.1
             NSAP 3911FF2299999900000000014900203599757201
```

Usage Notes

- Some ATMARP servers do not support PVC connections.

Context

- "ATMLIS Statement"
- "ATMPVC Statement" on page 56

ATMLIS Statement

Use the ATMLIS statement to define a logical IP subnetwork of an ATM network. Each LIS is a separate administrative entry and operates and communicates independently of other subnetworks on the same ATM network.

```
►►──ATMLIS──subnet_name──subnet_value──subnet_mask──────────────────────────►◄
```

Operands

subnet_name
The one- to 16-character name of this subnetwork.

subnet_value
The value (expressed in dotted-decimal form) produced when an IP address in the logical IP subnetwork is ANDed with the *subnet_mask*.

subnet_mask
A bit mask (expressed in dotted-decimal form) defining the subnet mask associated with the subnetwork. The bits should be contiguous in the mask and should include the high-order bits that comprise the network class mask.

Examples

1. This example shows an ATMLIS statement that defines a single subnetwork for an ATM network.

   ```
   ATMLIS ALLSUBNETS 125.0.0.0 255.0.0.0
   ```

2. This example shows ATMLIS statements that define three separate subnetworks for an ATM network.

   ```
   ATMLIS SUBNETA 125.1.0.0 255.255.0.0
   ATMLIS SUBNETB 125.2.0.0 255.255.0.0
   ATMLIS SUBNETC 125.3.0.0 255.255.0.0
   ```

Usage Notes

- An IP address is in a particular subnetwork if the result of ANDing it with the subnet mask is the subnet value.
- ATMLIS statements are referred to by ATMARPSERVER statements (see “ATMARPSERVER Statement” on page 54).
- Trailing zeros can be omitted from the *subnet_value* and *subnet_mask*.

Context

- “ATMARPSERVER Statement” on page 54

ATMPVC Statement

Use the ATMPVC statement to define a private virtual circuit for an ATM network.

```
►►──ATMPVC──PVC_name──link_name─────────────────────────────────────────►◄
```

Operands

PVC_name
: The one- to eight-character name of the PVC. This is the name used to define the PVC in the OSA configuration.

link_value
: The name of the ATM LINK (see “DEVICE and LINK Statement — ATM Devices” on page 64) with which this PVC is associated.

Usage Notes

- When an ATM device is started, all PVCs defined for its associated LINKs are started.
- ATMPVC statements may be referenced by ATMARPSERVER statements (see “ATMARPSERVER Statement” on page 54).

Context

- “ATMARPSERVER Statement” on page 54
- “DEVICE and LINK Statement — ATM Devices” on page 64

AUTOLOG Statement

The AUTOLOG statement identifies other virtual machines to be started by the TCPIP virtual machine when it begins execution.

The first AUTOLOG statement of a configuration file replaces the existing AUTOLOG list. Subsequent AUTOLOG statements in the same file add to the list.

```
                 ┌───────────────┐
                 ▼               │
►►──AUTOLOG────────user_id 0─────┴──ENDAUTOLOG─────────────────────────────►◄
```

Operands

user_id
: The name of a virtual machine that the TCPIP virtual machine should autolog. If a user ID is not valid, TCP/IP will display an error message.

0 A constant. For compatibility with prior releases of TCP/IP for VM, any arbitrary string is accepted.

Examples

This example shows how to include two servers in the AUTOLOG statement:

```
Autolog
    FTPSERVE 0         ; FTP Server
    NAMESRV  0         ; Domain Name Server
EndAutolog
```

Usage Notes

- The ENDAUTOLOG statement specifies the end of the AUTOLOG list.
- A virtual machine in the AUTOLOG list is terminated and restarted by TCP/IP whenever the stack is restarted. TCP/IP uses the CP FORCE command to terminate a virtual machine and the CP XAUTOLOG command to start a virtual machine.
- TCP/IP frequently checks to ensure that each virtual machine in the AUTOLOG statement is logged on.
- If the AUTOLOG list is empty or if the AUTOLOG statement is omitted, TCP/IP does not attempt to start any of the higher-level servers, such as FTP, MPROUTE, REXECD, or RouteD. Each server must be started manually.
- A virtual machine in the AUTOLOG list is expected to maintain its established connection to the stack. If this connection is terminated, the stack assumes a problem has occurred and attempts to reestablished the connection (by terminating and then restarting the virtual machine in question).
- If a virtual machine in the AUTOLOG list also has a PORT statement reserving a TCP or UDP port, but does not have a listening connection or is not accepting packets on that port, TCP/IP attempts to restart that virtual machine.

  If the PORT statement for a virtual machine specifies `NOAUTOLOG`, this restart will not be attempted. Please note that NOAUTOLOG will not prevent a restart of the virtual machine stemming from circumstances unrelated to the termination of the listening connection.

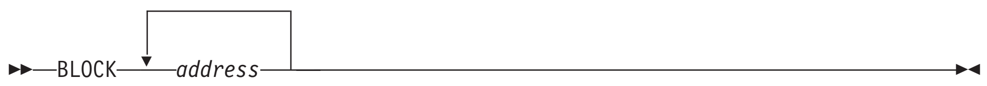
Context

- "PORT Statement" on page 117

BLOCK Statement

The BLOCK statement specifies IP addresses from which traffic is to be blocked. Any packets from blocked IP addresses are filtered out of the incoming data stream and are ignored.

The first BLOCK statement of a configuration file replaces the existing packet filters definitions. Subsequent BLOCK statements add filters to the existing list.



Operands

address
An IP address in dotted-decimal form. If the IP address of an incoming packet matches the address, the packet is filtered out. If a portion of the address is specified as an asterisk (*), the corresponding portion of the source address of the incoming packet is ignored.

Examples

- This example shows how to block packets from IP address 9.130.58.78:
```
BLOCK 9.130.58.78
```
- This example shows how to block packets 9.130.58.0 through 9.130.58.255:
```
BLOCK 9.130.58.*
```
- This example shows how to block packets from network 9 :
```
BLOCK 9.*
```

Usage Notes

- The NETSTAT command can be used to display and change the list of blocked IP addresses and to determine how many packets have been blocked.

BSDROUTINGPARMS Statement

If you are using ROUTED, use the BSDROUTINGPARMS statement to define the characteristics of all physical and virtual links.

```
►►──BSDROUTINGPARMS──┬─TRUE──┬──────────────────────────────────────────────►
                     └─FALSE─┘

   ┌────────────────────────────────────────────────────────┐
   ▼                                                        │
►────link_name──┬─DEFAULTSIZE─┬──metric subnet_mask dest_addr─┴──────────────►
                └─mtu─────────┘

►──ENDBSDROUTINGPARMS──────────────────────────────────────────────────────►◄
```

Operands

TRUE
Specifies that the maximum packet size for the interface is always used, regardless of the final destination host. This is the recommended setting.

FALSE
Specifies that the default maximum packet size of 576 is used when sending to networks that are not locally attached.

link_name
The name of the link, as defined in a LINK statement. Each link should be referenced once in the BSDROUTINGPARMS statement.

mtu
The maximum transmission unit (MTU) size in bytes for the network or host. The special value DEFAULTSIZE requests that TCP/IP supply a default value of 576. If you do not know the correct value to use, start by specifying DEFAULTSIZE as the packet size for each network and provide other values during later performance tuning. For IBM recommended maximum transmission unit definitions see the *max_packet_size* operand under the GATEWAY Statement on page 93.

metric
The metric associated with the cost of using the link. When it broadcasts routing information over this link, RouteD will add one more than the specified value to the metric for each route. If a metric of zero is specified, a value of one will be added, which is the default cost for a directly-connected network. If a metric of one is specified, a value of two will be added. As the metric gets higher, routes through the associated link become less preferred. The range is from 0 to 14. A metric of 0 is usually coded so that the route through this interface will be preferred to alternatives.

subnet_mask
A bit mask (expressed in dotted-decimal form) defining the subnet mask associated with the link. The bits must be contiguous in the network portion of the *subnet_mask*. **Unlike the GATEWAY statement, the high-order octets of the**

subnet mask that comprise the network ID must *not* be zero. If the *subnet_mask* is zero, RouteD will set the subnet mask to the network class mask.

Class A subnets must be 255.x.*y*.*z*, Class B subnets must be 255.255.*y*.*z*, and Class C subnets must be 255.255.255.*z*.

The topics ″Subnetting″ and ″Simplified IP Datagram Routing Algorithm with Subnets″ in the *TCP/IP Diagnosis Guide* illustrate the concept of subnetting and provide an example that describes how a subnet route is resolved. Subnets and subnet masks are also discussed in the IBM Redbook *IP Network Design Guide*, SG24–2580.

dest_addr

The address of the host on the other end of the link (applies to point-to-point links only). If the interface connects to a LAN network, specify 0 as the destination address. A non-zero destination address applies to non-broadcast and non-multicast-capable point-to-point links. For VIPA links, this field should be zero. The following device types are point-to-point: CLAW, CTC, SNALINK, IUCV, and PVMIUCV.

Examples

- This example shows the BSDROUTINGPARMS statement for several types of broadcast media, as well as for a point-to-point link to host 129.34.12.6.

```
BSDROUTINGPARMS FALSE

; Link Name  MTU          Metric   Subnet Mask     Destination Address
; ---------  -----------  -------  --------------  -------------------
;
  X25LA      1024         0        255.255.255.0   0
  ETH1       1500         0        255.255.255.0   0
  PCN1       2000         0        255.255.255.0   0
  TR1        2000         0        255.255.255.0   0
  HCH1       1018         0        255.255.255.0   0
  X25NPL1    DEFAULTSIZE  0        255.255.255.0   0
  TESTLINK   1500         0        255.255.0.0     129.34.12.6
  YORKTOWN   1500         0        255.0.0.0       0

ENDBSDROUTINGPARMS
```

- This example shows definitions for virtual devices.

```
BSDROUTINGPARMS false

; Link Name  MTU          Metric   Subnet Mask     Destination Address
; ---------  -----------  -------  --------------  -------------------
;
  VLINK1     DEFAULTSIZE  0        255.255.255.252 0
  VLINK2     DEFAULTSIZE  0        255.255.255.252 0

ENDBSDROUTINGPARMS
```

- In the next example the BSDROUTINGPARMS statement has been configured to define two interface links; for each link, a variable-length subnet mask is employed.

```
ASSORTEDPARMS
  VARSUBNETTING
  IGNOREREDIRECT
ENDASSORTEDPARMS

DEVICE 2216TR LCS          2900
LINK TR2216 IBMTR     0 2216TR
```

```
DEVICE ILANS1 ILANS       804
LINK TR1 IBMTR        0 ILANS1

HOME
  9.130.248.50    TR2216
  9.130.48.66     TR1

BSDROUTINGSPARMS FALSE
 ; Link Name MTU      Metric   Subnet Mask     Destination Addr
 ; --------- -------- -------- --------------- ----------------
   TR2216     2000     0       255.255.255.240  0
   TR1        2000     0       255.255.255.0    0
ENDBSDROUTINGPARMS

START 2216TR
START ILANS1
```

Usage Notes

- Use the BSDROUTINGPARMS statement ***only*** if you are running the RouteD server. If you are not running RouteD, this statement is ignored; use the GATEWAY statement instead.
- If a BSDROUTINGPARMS statement is encountered during Obeyfile processing while an MPROUTE dynamic server is running, an error message will be returned and the statement will be ignored.
- RouteD obtains the characteristics associated with each interface or link (defined using BSDROUTINGPARMS) from the TCP/IP stack. RouteD uses this information when it constructs routing table entries as well as when it updates the IP routing table used by the TCP/IP stack.
- When using the OBEYFILE command to modify the BSDROUTINGPARMS statement, keep these rules in mind:
  – If you change the MTU size of an existing link, issue the SMSG command RECONFIG for the RouteD server.
  – If you add new links, the RouteD server must be restarted. Also, it is recommended that you provide a HOME statement for each new link.
  – If you change the subnet mask of an existing link, include an empty GATEWAY statement to force TCP/IP to reinitialize its routing tables.
- The MTU value specified on the BSDROUTINGPARMS statement will also be used for applications that use the `setsockopt()` IP_MULTICAST_IF option to specify the route for multicast datagrams
- The RouteD server does not have to be restarted if you have added new links in the BSDROUTINGPARMS statement using the OBEYFILE command. When issuing the OBEYFILE command, include both the HOME and BSDROUTINGPARMS statements. To change BSDROUTINGPARMS for links already in use by RouteD, the RouteD RECONFIG SMSG command must be issued following the OBEYFILE command for the modified HOME and BSDROUTINGPARMS statements.
- RouteD will override any routes that are defined in a GATEWAY statement.
- Static routes should be defined as `passive` in the ETC GATEWAYS file used by RouteD.
- Rules for defining virtual IP addresses for virtual links are explained in "HOME Statement" on page 100.
- When RouteD broadcasts the route for the VIPA network the metric of the primary interface is used.

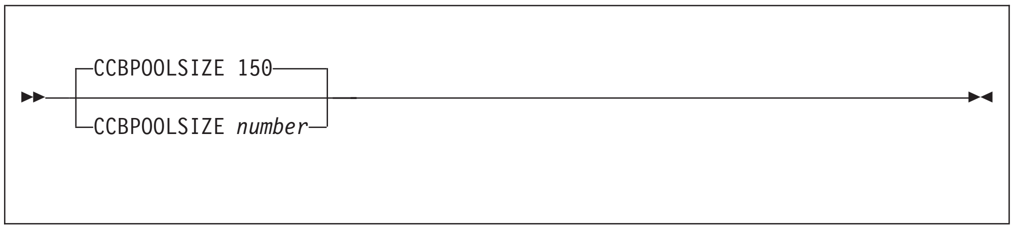
See “Chapter 7. Configuring the ROUTED Server” on page 155 for details.

Context

- “DEVICE and LINK Statement — Virtual Devices (VIPA)” on page 86
- “GATEWAY Statement” on page 92
- “HOME Statement” on page 100
- “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135
- “Chapter 7. Configuring the ROUTED Server” on page 155

CCBPOOLSIZE Statement

Use the CCBPOOLSIZE statement to set the initial number of client control blocks (CCBs). A CCB is needed for each virtual machine using the TCPIP virtual machine, including servers. Each one requires 280 bytes.



Operands

number
The initial number of CCBs in the free pool. The default is 150.

Examples

This example shows a CCBPOOLSIZE statement that defines the number of CCBs to be the default of 150.

```
CCBpoolSize   150
```

Usage Notes

- When running with ASSORTEDPARMS PERMITTEDUSERSONLY, every user mentioned in the configuration file uses a CCB. Otherwise, only the users mentioned in an OBEY or INFORM statement use a CCB. The CCB is used even if the user is not actively using TCP/IP services.
- The system will attempt to dynamically allocate 10% more CCBs any time the CCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many CCBs your system is using. To avoid dynamic allocation of CCBs during operation, use the CCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- “INFORM Statement” on page 103
- “OBEY Statement” on page 115
- *TCP/IP User’s Guide* for the NETSTAT command

DATABUFFERLIMITS Statement

Use the DATABUFFERLIMITS statement to specify the maximum number of data buffers that may be allocated for a TCP connection that is using window scaling.

```
                   ┌─5──────────┐  ┌─5─────────────┐
►►──DATABUFFERLIMITS──┴─send_limit─┴──┴─receive_limit─┴──────────────►◄
```

Operands

send_limit
: The maximum number of buffers to hold outbound data.

receive_limit
: The maximum number of buffers to hold inbound data. This limit, when multiplied by the regular data buffer size (see "DATABUFFERPOOLSIZE Statement") determines the maximum receive window size that is advertised.

Usage Notes

- This statement only affects connections with other hosts that also support RFC 1323.

Context

- "DATABUFFERPOOLSIZE Statement"

DATABUFFERPOOLSIZE Statement

Use the DATABUFFERPOOLSIZE statement to set the initial number and size of regular data buffers. Regular data buffers are used by the TCP layer for connections.

```
    ┌─DATABUFFERPOOLSIZE 160 8192──────────┐
►►──┼──────────────────────────────────────┼──────────────────►◄
    │                            ┌─8192─┐  │
    └─DATABUFFERPOOLSIZE number──┴─size─┴──┘
```

Operands

number
: The initial number of regular data buffers in the free pool. The default is 160.

size
: The size of each regular data buffer (in bytes). The default size is 8192.

  Only one of the following values or their alternates (shown in parenthesis) can be specified:

8192 (8K)	49152 (48K)
12288 (12K)	65536 (64K)
16384 (16K)	98304 (96K)
24576 (24K)	131072 (128K)
28672 (28K)	196608 (192K)
32768 (32K)	262144 (256K)

If you are using Offload, RISC System/6000 IP connections, or other CLAW connections, you must specify 28672 or 28K. Otherwise, a value of 32768 or 32K will optimize data transfer for FTP clients that support window sizes greater than 8,192.

Examples

- This example shows a DATABUFFERPOOLSIZE statement that defines the default of 160 buffers, each 8,192 bytes in length.

```
DataBufferPoolSize   160   8192
```

- This example shows a DATABUFFERPOOLSIZE statement for Offload that defines 160 data buffers, each 28,672 bytes in length.

```
DataBufferPoolSize   160   28K
```

Usage Notes

- The system will attempt to dynamically allocate 10% more regular data buffers any time the regular data buffer free pool level drops below 5%. You can use the NETSTAT POOLSIZE command to monitor how many regular data buffers your system is using. To avoid dynamic allocation of regular data buffers during operation, use the DATABUFFERPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- The TCP layer for Telnet uses regular data buffers only if there are no small data buffers available. The Telnet server also uses regular data buffers for internal processing, regardless of whether small data buffers are available.
- Increasing the size of regular data buffers will usually improve FTP throughput significantly.
- Regular data buffers are used by TCP/IP to communicate with Offload hosts.

Context

- "DATABUFFERLIMITS Statement" on page 63
- "SMALLDATABUFFERPOOLSIZE Statement" on page 124
- "TINYDATABUFFERPOOLSIZE Statement" on page 128
- *TCP/IP User's Guide* for the NETSTAT command

DEVICE and LINK Statement — ATM Devices

Use the DEVICE statement to specify the name and address of each Open Systems Adapter-2 (OSA-2) device that is configured for HPDT ATM native mode.

Use the LINK statement to define the ATM interface to the network.

```
►►──DEVICE──device_name──ATM──device_addr──PORTNAME port_name──────────────►◄
```

Operands

device_name
A unique name for the device. The maximum length is 16 characters. This name is referenced in the LINK statement.

ATM
Specifies that the device is an ATM device.

device_addr
The hexadecimal device address of the Open Systems Adapter (OSA). TCP/IP uses *device_addr* and *device_addr*+1.

PORTNAME *port_name*
The one- to eight-character *port_name* specifies the port name associated with the OSA device in the OSA configuration.

```
►►──LINK──link_name──ATM──device_name──────────────────────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

ATM
Specifies that the link is a ATM network connection.

device_name
The *device_name* must be the one specified in the DEVICE statement.

Examples

- In this example, OSA1 is an OSA-2 configured for HPDT ATM native mode with an attachment to an ATM network. The network has no subnetworks and has an ATMARP server at IP address 125.0.0.1. The VM host’s IP address is 125.0.0.44.

```
DEVICE OSA1   ATM 1E04
LINK   ATM1   ATM      OSA1
ATMLIS AllSubnets 125.0.0.0 255.0.0.0
ATMARPServer Server1 AllSubnets
             SVC  125.0.0.1
             NSAP 3911FF2299999900000000014900203599750501
HOME 125.0.0.44 ATM1
```

Usage Notes

- Do not use ATM DEVICE and LINK statements for OSA-2 devices operating in LAN emulation mode. Instead, use LCS DEVICE and LINK statements, as described in “DEVICE and LINK Statement — LCS Devices” on page 73.

Context

- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

DEVICE and LINK Statement — CTC Devices

Use the DEVICE statement to specify the name and device address of each channel-to-channel (CTC) device that you use.

Use the LINK statement to define the point-to-point network interface to the remote host.

```
►►──DEVICE──device_name──CTC──device_addr──────────────────────────────────►◄
```

Operands

device_name
A unique name for 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 device.

device_addr
The hexadecimal device address associated with the CTC adapter. TCP/IP uses
| *device_addr* and *device_addr*+1. If you are using HCD on OS/390, these devices
| must be defined as "CTCA" on the VM side, and "BCTC" on the OS/390 side.

```
►►──LINK──link_name──CTC──adapter_number──device_name──────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

CTC
Specifies that the link is a channel-to-channel adapter.

adapter_number
Must be '0' or '1'. Used to specify which device address is the read address and which is the write address. Use '0' to indicate that *device_addr* is read, and '1' to indicate *device_addr* is write.

device_name
The *device_name* must be the one specified in the DEVICE statement.

Context

- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135
- "Point-to-Point Connections to Other Hosts" on page 43

DEVICE and LINK Statement — HYPERchannel A220 Devices

Use the DEVICE statement to specify the name and device address of each HYPERchannel A220 device that you use.

Use the LINK statement to define the network interface for each HYPERchannel A220 device.

```
►►──DEVICE──device_name──HCH──device_addr──────────────────────────────►◄
```

Operands

device_name
A unique name for 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.

device_addr
The hexadecimal device address associated with the A220 adapter. TCP/IP uses *device_addr* and *device_addr*+1. These addresses must be defined to CP as type UNSUPPORTED.

```
►►──LINK──link_name──HCH──adapter_number──device_name──────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

HCH
Specifies that the link is a HYPERchannel A220.

adapter_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 one specified in the DEVICE statement.

Usage Notes

- Because the ATTENTION+BUSY and unit check conditions are normally handled in the background, they can affect performance without any visible evidence. The recommended 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 TCP/IP avoids a 10 millisecond delay when retrying a command that produced a unit check. This setting also eliminates the need for TCP/IP to perform sense operations.
- You can specify TRACE HCH in the configuration file to have TCP/IP display all ATTENTION+BUSY and unit check conditions, even those from which it recovers. This allows you to compare the effects of different mode switch settings.

Context

- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

DEVICE and LINK Statement — Integrated Communication Adapters

Use the DEVICE statement to specify the name and device address of each integrated communications adapter (ICA) you use.

Use the LINK statement to define the network interface for each Ethernet, IBM Token-Ring, or X.25 network to which the ICA is attached.

```
►►──DEVICE──device_name──┬─ELANS──┬──device_addr──────────────────────────────►
                         ├─ILANS──┤
                         └─X25ICA─┘

►─┬──────────────────────────────────────────────────────────────────┬──────►◄
  └─wait_time──┤ Optional Parameters 1 ├──┤ Optional Parameters 2 ├─┘
```

Optional Parameters 1:

```
   ┌─20─────┐  ┌─5─────────────┐  ┌─20───────────┐  ┌─20──────────┐
├──┴─in_buf─┴──┴─in_threshold──┴──┴─in_gap_timer─┴──┴─in_msg_recv─┴──────────┤
```

Optional Parameters 2:

```
   ┌─20──────┐  ┌─17────────────┐  ┌─200───────────┐  ┌─200───────────┐
├──┴─out_buf─┴──┴─out_threshold─┴──┴─out_gap_timer─┴──┴─out_msg_timer─┴──────┤
```

Operands

device_name
A unique name for the device. The maximum length is 16 characters.

For X25ICA devices, the name must be unique within the first 8 characters. See the Usage Notes section for details as it is used on the SETNET command.

ELANS
Specifies that the device is an Ethernet Protocol Network LAN Subsystem adapter.

ILANS
Specifies that the device is an IBM Token-Ring LAN Subsystem adapter.

X25ICA
Specifies that the device is an X.25 Communications Subsystem adapter.

device_addr
The hexadecimal device address associated with the adapter. TCP/IP uses *device_addr* through *device_addr*+3. These device addresses must be defined to CP as type ICA_ETHERNET, ICA_TOKENRING, or ICA_HDLC.

wait_time
Controls the amount of channel bandwidth used for the control port. If there is no activity on the data ports, the adapter pauses after every CWRITE statement for the number of seconds specified. This pause allows other adapters to access the channel. The default value is 0.

This value is ignored on ELANS devices.

in_buf
The number of buffers to hold inbound data. The default value is 20. You should increase the default value if there is evidence that packets are being dropped.

in_threshold
The number of inbound data buffers that can be transferred before an ATTENTION interrupt is presented to the interrupt port. The threshold value should be less than 7. The default value is 5.

in_gap_timer
The amount of time the adapter waits after transferring a burst of inbound data buffers before presenting an ATTENTION interrupt to the interrupt port. The default value is 20 units, where each unit is 64 microseconds.

in_msg_recv
The amount of time the adapter waits after transferring a burst of inbound data buffers. The default value is 20 units, where each unit is 64 microseconds.

This value is ignored for ILANS and ELANS devices.

out_buf
The number of buffers to hold outbound data. The default value is 20.

out_threshold
The number of outbound data buffers that can be transferred before presenting an ATTENTION interrupt to the interrupt port. The threshold value should be approximately the same as *out_buf*. The default value is 17.

out_gap_timer
The amount of time the adapter waits after transferring a burst of outbound data buffers before presenting an ATTENTION interrupt to the interrupt port. The default value is 200 units, where each unit is 64 microseconds.

out_msg_recv
The amount of time the adapter waits after transferring a burst of outbound

data buffers before presenting an ATTENTION interrupt to the receiving side. The default value is 200 units, where each unit is 64 microseconds.

This value is ignored for ILANS and ELANS devices.

```
►►──LINK──link_name──protocol──link_number──device_name──────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

protocol
The LAN protocol to be supported for this device.

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.

IBMTR
IBM Token-Ring protocol only

X25ICA
X.25 protocol only

link_number
A unique integer that identifies the port on the ICA. Because each ICA has only one port, the *link_number* is ignored, but must be specified to serve as a placeholder.

device_name
The *device_name* must be the one specified in the DEVICE statement.

Usage Notes

- Each X25ICA device requires additional configuration using the SETNET *device_name* command. When the device is started, TCP/IP will search for a file with a file name of *device_name* and a file type of X25ICA, containing the parameters of the particular X.25 adapter. Only the first 8 characters of the device name are used to construct the file name. Thus, if several adapters are controlled by the same TCPIP virtual machine, their device names must be unique in the first 8 characters.
- The integrated adapters in IBM 9370 and 9221 processors may not exhibit the performance characteristics necessary to support all TCP/IP application protocols. For example, NFS can overrun the Ethernet ICA.

Context

- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

DEVICE and LINK Statement — Local IUCV Connections

Use the DEVICE statement to define an IUCV connection to another virtual machine on the same VM system that is running a TCP/IP for VM stack.

Use the LINK statement to define the point-to-point network interface to another TCP/IP stack.

The other TCP/IP stack must have a corresponding pair of DEVICE and LINK statements.

```
►►──DEVICE──device_name──IUCV──0 0──other_virtual_machine──priority──────────►◄
```

Operands

device_name
A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

IUCV 0 0
Specifies that the device is using an IUCV connection.

other_virtual_machine
The name of another virtual machine running TCP/IP for VM to which you want to establish a connection.

priority
The order of priority between the two connected virtual machines. Use A on one virtual machine and B on the other.

```
►►──LINK──link_name──IUCV──link_number──device_name──────────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

IUCV
Specifies the device is using an IUCV connection.

link_number
Must be an integer, but its value is ignored. This parameter is included for consistency with LINK statements for other device types.

device_name
: The *device_name* must be the one specified in the DEVICE statement.

Context

- “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135
- “Point-to-Point Connections to Other Hosts” on page 43

DEVICE and LINK Statement — Remote IUCV Connections

Use the DEVICE statement to define an IUCV connection to a virtual machine on a remote VM system that is running the TCP/IP for VM stack.

The connection is established using the Personal Computer Communications Facility (PCCF) of VM/Pass-Through Facility Version 1 Release 4 or later. This facility is usually referred to as “PVM IUCV”.

Use the LINK statement to define the point-to-point network interface to the remote TCP/IP stack.

The remote TCP/IP stack must have a corresponding pair of DEVICE and LINK statements.

```
►►──DEVICE──device_name──PVMIUCV──rmt_pvm_node──rmt_tcpip_vmid──────────────►

►─local_pvm_vmid──local_pvm_node─────────────────────────────────────────────►◄
```

Operands

device_name
: A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

PVMIUCV
: Specifies that the device connection is to another VM system using PVM IUCV.

rmt_pvm_node
: The PVM network node name of the remote node.

rmt_tcpip_vmid
: The name of the virtual machine of the TCPIP virtual machine on the remote node.

local_pvm_vmid
: The name of the virtual machine of the PVM server on the local node.

local_pvm_node
: The PVM network node name of the local node.

```
►►──LINK──link_name──IUCV──link_number──device_name─────────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

IUCV
Specifies the device is using an IUCV connection.

link_number
Must be an integer, but its value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name
The *device_name* must be the one specified in the DEVICE statement.

Context

- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135
- "Point-to-Point Connections to Other Hosts" on page 43

DEVICE and LINK Statement — LCS Devices

Use the DEVICE statement to specify the name and device address of each Open Systems Adapter in LAN Emulation mode, IBM 8232 LAN channel station (LCS), or IBM 3172 Interconnect Controller that you use.

Use the LINK statement to define the network interface for each Ethernet, IBM Token-Ring, PC Network, or FDDI network to which the LCS is attached.

There can be as many LINK statements for each LCS as there are LAN interfaces in the LCS.

```
►►──DEVICE──device_name──LCS──device_addr──┬────────┬──────────────────►◄
                                           └─NETMAN─┘
```

Operands

device_name
A unique name for 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_addr
The hexadecimal address of the LCS. TCP/IP uses *device_addr* and *device_addr*+1.

NETMAN
Specifies that this device is an IBM 3172 that supports the IBM Enterprise-specific MIB variables for the 3172. These variables can be retrieved by SNMP.

LINK Statement for Ethernet Network LCS

This LINK statement is used to define the Ethernet adapter on an LCS previously defined by an LCS DEVICE statement.

```
►►──LINK──link_name──ETHERNET──link_number──device_name──────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

ETHERNET
Specifies that the link is to an Ethernet network.

link_number
The relative Ethernet adapter number within the LCS. This number is assigned by the Interconnect Controller Program (ICP).

device_name
The *device_name* must be the same name as specified in the DEVICE statement.

LINK Statement for Token-Ring Network or PC Network LCS:

This LINK statement is used to define the IBM token-ring adapter on an LCS previously defined by an LCS DEVICE statement. The IBM token-ring LINK statement is also used to define a 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 TCP/IP 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.

```
                                                   ┌─NONCANONICAL─┐
►►──LINK──link_name──IBMTR──link_number──device_name─┼──────────────┼───────────►
                                                   └─CANONICAL────┘

  ┌─ALLRINGSBCAST─┐
►─┼───────────────┼──────────────────────────────────────────────────────────►◄
  └─LOCALBCAST────┘
```

Operands

link_name
: A unique name for the link. The maximum length is 16 characters.

IBMTR
: Specifies that the link is to an IBM Token-Ring network.

link_number
: The relative IBM Token-Ring adapter number within the LCS. This number is assigned by the Interconnect Controller Program (ICP).

device_name
: The *device_name* must be the one specified in the DEVICE statement.

NONCANONICAL
: MAC addresses in ARP packets on this token-ring network are in the more common non-canonical format. This is the default.

CANONICAL
: MAC addresses in Address Resolution Protocol (ARP) packets on this token-ring network are in the canonical IEEE 802.5 form.

ALLRINGSBCAST
: All IP and ARP broadcasts are sent as all-rings broadcasts, which are propagated through token-ring bridges. 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.

LINK Statement for FDDI LCS:

This LINK statement is used to define the Fiber Distributed Data Interface (FDDI) adapter on an LCS previously defined by an LCS DEVICE statement.

```
►►──LINK──link_name──FDDI──link_number──device_name──────────────────────────►◄
```

Operands

link_name
: A unique name for the link. The maximum length is 16 characters.

FDDI
Specifies that the link is to a FDDI network.

link_number
The relative FDDI adapter number within the LCS. This number is assigned by the Interconnect Controller Program (ICP).

device_name
The *device_name* must be the one specified in the DEVICE statement.

Examples

- In this example, LCS2 is an IBM 3172 Model 2 with one IBM Token-Ring and one Ethernet adapter.

```
DEVICE LCS2   LCS           BA0
LINK TR1   IBMTR    0 LCS2
LINK ETH1  ETHERNET 0 LCS2
```

- In this example, LCS3 is an IBM 3172 Model 3 with a single FDDI adapter and that supports management by SNMP.

```
DEVICE LCS3   LCS           BE0  NETMAN
LINK FDDI1 FDDI     0 LCS3
```

- This example shows how you might code DEVICE, LINK, and related statements for an IBM 3172 Model 3 that has one Ethernet and two IBM Token-Ring adapters.

```
DEVICE LCS3   LCS             BA0 NETMAN
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

; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------
;
  192.10.10     =            TR1       2000         0
  9             =            TR1       2000         0.0.255.0   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 RouteD
;
; BSDROUTINGPARMS FALSE
;
; ; Link Name  MTU          Metric   Subnet Mask     Destination Address
; ; ---------  -----------  -------  --------------  -------------------
; ;
;   TR1        2000         0        255.255.255.0   0
;   TR2        2000         0        255.255.255.0   0
;   ETH1       1500         0        255.255.255.0   0
;
; ENDBSDROUTINGPARMS
;

START LCS1
```

Context

- "BSDROUTINGPARMS Statement" on page 59
- "GATEWAY Statement" on page 92
- "HOME Statement" on page 100
- "START Statement" on page 125
- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

DEVICE and LINK Statement — Offload Hosts

Use the DEVICE statement to specify the name and device address of each Offload host that you use.

Use the LINK statement to define network interfaces associated with each Offload device. The DEVICE statement is followed by at least two LINK statements:
- One IP LINK statement
- One or more API LINK statements

If you use these statements, you should also include SMALLDATABUFFERPOOLSIZE and TINYDATABUFFERPOOLSIZE statements in your configuration.

```
►►──DEVICE──device_name──CLAW──device_addr──host_claw_name──────────────────►

►─workstation_claw_name──NONE─────────────────────────────────────────────►◄
```

Operands

device_name
: A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CLAW
: Specifies that the device uses the CLAW channel protocol.

device_addr
: The hexadecimal device address of the Offload host. TCP/IP uses *device_addr* and *device_addr*+1.

  CLAW devices must be defined in IOCP as device type 3088. For more information on defining devices with IOCP, see *Input/Output Configuration Program User Guide*, GC38-0401.

  A device definition is not required for CLAW devices on z/VM. For more information on when and how devices are defined for z/VM, see *z/VM: Planning and Administration*.

TCPIP
: A constant specific to the TCP/IP for VM Offload feature.

OS2TCP
: A constant specific to the TCP/IP for VM Offload feature.

NONE
A placeholder reserved for future use.

If offload services are being provided by other than the TCP/IP for VM Offload feature, the values specified on the DEVICE statement may be different than those listed here. Consult the offload product's documentation for details.

IP LINK Statement:

The LINK statement is used to define the link between the IP layer of TCP/IP on VM and the IP layer on the Offload host.

```
►►──LINK──ip_link_name──OFFLOADLINK1──1──device_name──────────────────────────►◄
```

Operands

ip_link_name
A unique name for the IP link. The maximum length is 16 characters.

OFFLOADLINK1
Specifies that the link is the IP link to the Offload host.

1 A constant.

device_name
The *device_name* must be the one specified in the DEVICE statement.

API LINK Statement:

The LINK statement used to specify each network interface on the Offload host. Specify one API LINK for each network interface except the IP LINK to TCP/IP for VM.

```
►►──LINK──link_name──link_type──internet_addr──device_name──ip_link_name──────►◄
```

Operands

link_name
A unique name for the API link. The maximum length is 16 characters.

link_type
Specifies the general category of the network. The *link_type* is used to determine the flags that are returned by the SIOCGIFFLAGS option of the ioctl() API. There are three categories:

OFFLOADAPIBROAD
Networks that support broadcast, such as Ethernet, Token-Ring, and FDDI (Returns IFF_BROADCAST, IFF_NOTRAILERS)

OFFLOADAPIPP
Point-to-point networks (Returns IFF_POINTOPOINT)

OFFLOADAPIOTHER
Other kinds of networks (Returns no additional flags)

See *TCP/IP Programmer's Reference* for additional API programming information.

internet_addr
The dotted-decimal interface address (home address) of the Offload host interface to this network.

device_name
The *device_name* must be the one specified in the DEVICE statement.

ip_link_name
The *ip_link_name* must be the one specified in the associated IP LINK statement.

Examples

- In this example, OFF1 is an IBM 3172 Model 3 with one LAN adapter running Offload.

```
DEVICE OFF1 CLAW 5A0 TCPIP OS2TCP NONE
LINK IPLINK1 OFFLOADLINK1 1 OFF1
LINK OFFLAN1 OFFLOADAPIBROAD 9.67.43.200 OFF1 IPLINK1
```

- This example shows how you might code DEVICE, LINK, and related statements for an Offload host connection.

```
DEVICE OFF1 CLAW 5A0 TCPIP OS2TCP NONE
LINK IPLINK1 OFFLOADLINK1 1 OFF1
LINK OFFTR1  OFFLOADAPIBROAD 192.10.10.10 OFF1 IPLINK1
LINK OFFETH1 OFFLOADAPIBROAD 128.50.17.1  OFF1 IPLINK1

PRIMARYINTERFACE OFFTR1

GATEWAY
;
; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------

  192.10.10     =            OFFTR1    2000         0
  128.50        =            OFFETH1   1500         0.0.240.0    0.0.16.0
  DEFAULTNET    192.10.10.1  OFFTR1    DEFAULTSIZE  0

; The following BSDROUTINGPARMS statement would be used if running RouteD
;
; BSDROUTINGPARMS FALSE
;
; ; Link Name  MTU          Metric   Subnet Mask     Destination Address
; ; ---------  -----------  -------  --------------  -------------------
; ;
;   OFFTR1     2000         0        255.255.255.0   0
;   OFFETH1    1500         0        255.255.240.0   0
;
; ENDBSDROUTINGPARMS
;

START OFF1
```

Usage Notes

- You cannot add new DEVICE and LINK statements for Offload devices using the OBEYFILE command.
- The offloading of TCP/IP processing for VM uses the common link access to workstation (CLAW) protocol for communicating between the Offload host and the TCPIP virtual machine. CLAW is a low-level protocol linking the TCPIP virtual machine and the Offload server in the Offload host. CLAW provides a reliable connection protocol between two applications.
- CLAW uses two System/® subchannels to communicate with the Offload host. One subchannel is used for all data from the VM host to the Offload host (writes), the other subchannel is used for all data from the Offload system to the VM host (reads). CLAW can multiplex up to 31 logical links over a pair of subchannels, but only two CLAW logical links are used to communicate between TCP/IP for VM and the Offload host:
  1. A link carrying IP datagrams between the IP layer of TCP/IP for VM and the IP layer of TCP/IP for Offload. This link is generally used only when routing datagrams between a network interface on the Offload host and a conventional network interface on TCP/IP for VM.
  2. A link carrying socket application programming interface requests between the Pascal and socket API layers of TCP/IP for VM and the Offload server on the Offload host.

Context

- “BSDROUTINGPARMS Statement” on page 59
- “GATEWAY Statement” on page 92
- “PRIMARYINTERFACE Statement” on page 119
- “SMALLDATABUFFERPOOLSIZE Statement” on page 124
- “START Statement” on page 125
- “TINYDATABUFFERPOOLSIZE Statement” on page 128
- “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135

DEVICE and LINK Statement — OSD Devices

Use the DEVICE statement to specify the name and address of the device that will use the Queued Direct I/O Hardware Facility (QDIO).

Use the LINK statement to define the network interface for each OSD device.

```
►►──DEVICE──device_name──OSD──device_addr──PORTNAME port_name──────────────────►

                          ┌─NONRouter──┐  ┌─NOAUTORestart─┐
►──┬─────────────────┬────┼────────────┼──┼───────────────┼──────────────────►◄
   └─PORTNUmber n────┘    ├─PRIRouter──┤  └─AUTORestart───┘
                          └─SECRouter──┘
```

Operands

device_name
: A unique name for the device whose maximum length is 16 characters. This name is referenced in the LINK statement.

OSD
: Specifies that the device is an OSA Express Adapter using the QDIO Hardware Facility.

device_addr
: A hexadecimal device address that specifies the first of three consecutive virtual device numbers to be grouped for the OSA Express Adapter. TCP/IP uses *device_addr* and *device_addr*+1 and *device_addr*+2.

PORTNAME *port_name*
: The eight character name used to identify the OSA Express Adapter. See the "Usage Notes" on page 82 for details.

PORTNUmber *n*
: This operand maps a port number to the PORTNAME specified. If not defined, the PORTNumber defaults to 0. For Giga Ethernet, this operand is ignored.

NONRouter
: If a datagram is received at this device for an unknown IP address, the datagram will be discarded instead of rerouted.

PRIRouter
: If a datagram is received at this device for an unknown IP address, the datagram will be routed to the TCP/IP instance to which this device was defined with the PRIRouter parameter.

SECRouter
: If a datagram is received at this device for an unknown IP address, and the TCP/IP connection defined as the primary router is not active, then the datagram will be routed to the TCP/IP instance to which this device was defined with the SECRouter parameter.

AUTORestart
: Instructs the TCP/IP server to attempt to restart the device in the event of a device failure. AUTORestart will only be attempted after successful data transfer has occurred.

NOAUTORestart
: Instructs the TCP/IP server to not attempt to restart the device. By default, TCP/IP does not restart devices that have failed.

```
►►──LINK──link_name──┬─QDIOATM───────┬──device_name──────────────────────────►◄
                     └─QDIOETHERNNET─┘
```

Operands

link_name
: A unique name for the link. The maximum length is 16 characters. The first eight characters of the *link_name* must be unique because it identifies a specific connection for the stack.

QDIOATM or QDIOETHERNET
Specifies that the link is either an ETHERNET or ATM connection on a Queued Direct I/O Hardware Facility.

device_name
The *device_name* must be the one specified in the DEVICE statement.

Examples

In this example, QDO1 is an OSA Express Ethernet adapter that is configured to use the Queued Direct I/O Hardware Facility. The home IP address of the z/VM host is 125.0.0.33.

```
DEVICE QDO1    OSD 1D01 PORTNAME BIGANG
LINK   GIG1  QDIOETHERNET   QDO1
HOME  125.0.0.33 GIG1
TCP/IP is using device address 1D01-->1D03
```

Usage Notes

- When the device is started, TCP/IP will assign the PORTNAME as the hardware adapter name. If an adapter name was already assigned by a previous connection, then the PORTNAME must be the same as assigned by any other connection in order to share the adapter. This includes sharing the OSA Express Adapter within this logical partition or all other partitions.
- The device address specified on the OSD device statement is the first of three consecutive device addresses to be grouped for the OSA Express adapter.
- To change the adapter name, the adapter must be reset. This can be accomplished by deconfiguring the CHPID for the OSA Express Adapter using either the HMC console or by issuing the VARY OFF CHPID command with the FORCE option. For more information on this command see the z/VM: CP Command and Utility Reference. If the OSA Express Adapter is shared by multiple logical partitions (LPARs), then you must deconfigure the CHPID from each partition sharing the adapter.

Context

- “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135

DEVICE and LINK Statement — RISC System/6000 IP Connections

Use the DEVICE statement to specify the name and device address of each RISC System/6000 that you use.

Use the LINK statement to define a point-to-point network interface to each RISC System/6000. Only one LINK statement should be used for each device.

```
►►──DEVICE──device_name──CLAW──device_addr──host──psca──NONE──────────────────►

  ┌─20───────────┐  ┌─20────────────┐  ┌─4096──────┐  ┌─4096───────┐
►─┴─read_buffers─┴──┴─write_buffers─┴──┴─read_size─┴──┴─write_size─┴──────────►◄
```

Operands

device_name
: A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statements.

CLAW
: A constant to state that this device is to run the CLAW protocol.

device_addr
: The hexadecimal address of the channel adapter in the RISC System/6000. TCP/IP uses *device_addr* and *device_addr*+1.

  These addresses must be defined to CP and the IOCP as 3088-type devices.

host
: A value that defines the name of the host system in the system validation exchange between TCP/IP for VM and the workstation. This host name either must be the same name that is defined as the CLAW-mode HOST name in SMIT on the RISC System/6000, or the default of **HOST**.

psca
: A value for the name of the workstation for the system validation exchange. This value either must be the name of the CLAW-mode adapter defined in SMIT on the RISC System/6000, or the default of **PSCA**.

NONE
: A placeholder reserved for future use.

read_buffers
: The number 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 to not affect overall system performance. The default is 20.

write_buffers
: The number 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 to not affect overall system performance. The default is 20.

read_size
: The size of the read buffers. Values are:

  1024

  2048

  3072

  4096

  It 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® attachments.

write_size
: The size of the write buffers. Values are:

  1024

  2048

  3072

4096

It 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 attachments.

```
►►──LINK──link_name──IP──0──device_name──────────────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

IP A constant.

0 A constant.

device_name
The *device_name* must be the one specified in the DEVICE statement.

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
;
; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        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 RouteD
;
; BSDROUTINGPARMS FALSE
;
; ; Link Name  MTU          Metric   Subnet Mask     Destination Address
; ; ---------  -----------  -------  --------------  -------------------
; ;
;   IPLINK1    2000         0        255.255.255.0   192.10.10.2
;
; ENDBSDROUTINGPARMS
;

START RS6K
```

Usage Notes

- If you are using CLAW device with TCP/IP for VM on a second-level VM system, the second-level TCPIP virtual machine must be running in Virtual=Real

(V=R) mode. This restriction is necessary to accommodate appropriate channel program translation and performance criteria.

Context

- "BSDROUTINGPARMS Statement" on page 59
- "GATEWAY Statement" on page 92
- "HOME Statement" on page 100
- "START Statement" on page 125
- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

DEVICE and LINK Statement — SNA LU0 Links

Use the DEVICE statement to define an SNA LU Type 0 connection to a remote VM or MVS system.

Use the LINK statement to define the point-to-point network interface to the remote system.

```
►►──DEVICE──device_name──SNAIUCV SNALINK──remote_lu_name──user_id──────────────►◄
```

Operands

device_name
: A unique name for the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

SNAIUCV SNALINK
: Specifies that the connection operates using SNA LU type 0.

remote_lu_name
: The logical unit (LU) name associated with the SNALINK program running on the remote VM or MVS system.

user_id
: The name of the virtual machine running the SNALINK program.

```
►►──LINK──link_name──IUCV──link_number──device_name────────────────────────────►◄
```

There must be only one LINK statement for each SNA LU type 0 device statement.

Operands

link_name
: A unique name for the link. The maximum length is 16 characters.

IUCV
: Specifies that the device is using an IUCV connection.

link_number
: Must be an integer, but its value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name
: The *device_name* must be the one specified in the DEVICE statement.

Examples

In this example, SNALU0 is a SNA Link.

```
DEVICE SNALU0 SNAIUCV SNALINK LU000000 SNALINK
LINK SNA1 IUCV     1 SNALU0
```

Usage Notes

- Both the local and the remote systems must be running the SNALINK application.
- You may specify the same SNALINK virtual machine on multiple SNALINK DEVICE statements. However, you must specify a different *remote_lu_name* for each DEVICE statement. This value is passed to the SNALINK application to establish a session with a remote LU of that name.
- Additional configuration of the SNALINK virtual machine is required.

Context

- "Point-to-Point Connections to Other Hosts" on page 43

DEVICE and LINK Statement — Virtual Devices (VIPA)

Use the DEVICE statement to specify the name and virtual address of a device.

Use the LINK statement to define the link that corresponds to a virtual device.

```
►►──DEVICE──device_name──VIRTUAL──0─────────────────────────────────────►◄
```

Operands

device_name
: A unique name for the device. The maximum length is 16 characters. This name is referenced in the LINK statement.

VIRTUAL
: Specifies that this device is not associated with real hardware and is used to provide fault tolerance. Virtual devices always stay active and are never subject to physical failure.

0 A constant.

```
►►──LINK──link_name──VIRTUAL──0──device_name──────────────────────────────────►◄
```

Operands

link_name
: A unique name for the link. The maximum length is 16 characters. The same name is specified in the HOME statement.

VIRTUAL
: Specifies that the link is virtual and is not associated with real hardware. It is used for fault tolerance support.

0 A constant.

device_name
: The device name must be the one specified on the DEVICE statement.

Examples

```
DEVICE VDEV1  VIRTUAL 0
LINK   VLINK1 VIRTUAL 0 VDEV1
DEVICE VDEV2  VIRTUAL 1
LINK   VLINK2 VIRTUAL 0 VDEV2
```

Usage Notes

- The term *virtual device* is used to describe a VIPA device; it is in no way related to VM's traditional virtual device support.
- Only one virtual link can be defined for a virtual device.
- More than one virtual DEVICE/LINK pair can be defined to provide multiple virtual IP addresses for a TCP/IP image.
- A virtual LINK cannot be coded on the START, GATEWAY, or TRANSLATE statements, but can be referenced on a BSDROUTINGPARMS statement to define interface characteristics such as a subnet mask.
- For rules about virtual IP address definitions for virtual links, see "HOME Statement" on page 100.

Context

- "BSDROUTINGPARMS Statement" on page 59
- "HOME Statement" on page 100
- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135
- "Point-to-Point Connections to Other Hosts" on page 43

DEVICE and LINK Statement — X.25 NPSI Connections

Use the DEVICE statement to specify the name and location of the X.25 NPSI interface program devices that you use.

Use the LINK statement to define a network interface for each network accessed by X.25 NPSI.

DEVICE and LINK: X.25 NPSI Connections

```
►►──DEVICE──device_name──X25NPSI──user_id──────────────────────────────────►◄
```

Operands

device_name
A unique name for 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 connection.

user_id
The name of the virtual machine that is running the X.25 NPSI server application, X25IPI.

Note: Only one DEVICE and one LINK statement are allowed per X25IPI virtual machine.

```
►►──LINK──link_name──IUCV──link_number──device_name────────────────────────►◄
```

Operands

link_name
A unique name for the link. The maximum length is 16 characters.

IUCV
Specifies that the device uses an IUCV connection.

link_number
Must be an integer, but its value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name
The *device_name* must be the one specified in the DEVICE statement.

Examples

This example shows how you might code DEVICE, LINK, and related statements for an X.25 connection.

```
DEVICE X25DEV X25NPSI X25IPI
LINK X25LINK IUCV 1 X25DEV
;
HOME
   199.005.058.23     X25LINK
;
GATEWAY
;
; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------
;
```

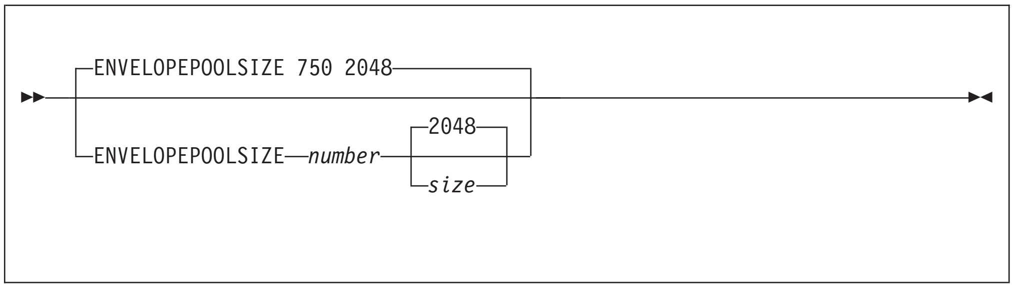
```
  192.005      =           X25LINK   2000        0.0.255.0   0.0.58.0
;
START X25DEV
;
```

Context

- “GATEWAY Statement” on page 92
- “HOME Statement” on page 100
- “START Statement” on page 125
- “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135

ENVELOPEPOOLSIZE Statement

Use the ENVELOPEPOOLSIZE statement to set the initial number of envelopes. Envelopes are used to hold datagrams and fragments of up to 2,048 bytes of data during TCP/IP processing.



Operands

number
: The initial number of envelopes in the free pool. The default is 750.

size
: The size of each envelope (in bytes); this *size* determines the maximum number of bytes that can be held by an envelope. The default is 2048. Only the following values or their alternatives (shown in parenthesis) may be specified:

    512

    1024 (1K)

    2048 (2K)

    4096 (4K)

    8192 (8K)

    16384 (16K)

    32768 (32K)

    65535 (64K)

Examples

This example shows an ENVELOPEPOOLSIZE statement that defines the number of envelopes to be the default of 750.

```
EnvelopePoolSize   750
```

Usage Notes

- Specify the *size* parameter to establish the size of the largest packet that can be sent and received.

  Note: Ensure the specified *size* does not exceed that defined for large envelopes (defined by the LARGEENVELOPEPOOLSIZE statement). Failure to do so will result in TCP/IP stack initialization errors.
- The system will attempt to dynamically allocate 10% more envelopes any time the envelope free pool level drops below 5%. You can use the NETSTAT POOLSIZE command to monitor how many small envelopes your system is using. To avoid dynamic allocation of small envelopes during operation, use the ENVELOPEPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- Running out of envelopes will result in lost packets.

Context

- "LARGEENVELOPEPOOLSIZE Statement" on page 110
- *TCP/IP User's Guide* for the NETSTAT command

FILE Statement

Use the FILE statement to specify a file to receive trace information.

```
►►──FILE──filename──filetype──filemode──────────────────────────────►◄
```

Operands

filename
The CMS file name of the file to receive the trace information.

filetype
The CMS file type of the file to receive the trace information.

filemode
The CMS file mode of the file to receive the trace information.

Usage Notes

- The current trace file is closed and the new file is opened as the trace file.
- If the specified trace file already exists, its previous contents are deleted.

Context

- "NOSCREEN Statement" on page 114
- "TRACE Statement" on page 130

FIXEDPAGESTORAGEPOOL

Use the FIXEDPAGESTORAGEPOOL statement to set the initial number of Fixed Page Storage Blocks (FPSBs) that will be preallocated and placed in this pool during initialization. FPSBs are 4k blocks (pages) of storage used by TCP/IP and the Queued Direct I/O Hardware Facility.

```
►►──┬──────────────────────────────────────────────────┬───────────►◄
    └─FIXEDPAGESTORAGEPOOL──fpsbs──┬──────────────┬──┘
                                   └─fpsb_limit─┘
```

Operands

fpsbs
: The initial number of FPSBs to be preallocated in the pool. If the value specified for *fpsbs* is zero or the FIXEDPAGESTORAGEPOOL Statement was not specified, then no storage will be preallocated. The FIXEDPAGESTORAGEPOOL will be populated ″as needed″ based on network traffic, until the maximum allowable number (*fpsb_limit*) of FPSBs are allocated or until 90% of available virtual machine storage has been consumed.

fpsb_limit
: The maximum number of FPSBs that may be allocated for this virtual machine. This value must be equal to or greater than the value specified by *fpsbs.*

Examples

This example shows a FIXEDPAGESTORAGEPOOL statement that defines a storage cap of 2000 pages (8 MB). No pages will be allocated for the storage pool at TCP/IP initialization. FPSBs are allocated, as network activity dictates, up to the specified *fpsb_limit*.

```
FIXEDPAGESTORAGEPOOL 0 2000
```

This example shows a FIXEDPAGESTORAGEPOOL statement which defines 1500 pages (6 MB) to be allocated at TCP/IP initialization. Since *fpsb_limit* is not specified, the storage pool will default to using up to 90% of the available CMS storage within the TCP/IP virtual machine.

```
FIXEDPAGESTORAGEPOOL 1500
```

Usage Notes

- By default, the TCP/IP server manages the storage pool to utilize up to 90% of the available CMS storage within the TCPIP virtual machine. If this default does not meet your needs or you require a specific amount of storage to be allocated at TCP/IP initialization, specify your required storage allocation by using the FIXEDPAGESTORAGEPOOL statement. For each DEVICE OSD statement that is active, 1000 to 1500 FPSBs (4 to 6 MB) are recommended.
- When the number specified by the *fpsb_limit* exceeds the available CMS free storage in the TCPIP server virtual machine, TCP/IP will recalculate the FPSB_LIMIT to be the total available CMS free storage, maintaining at least 10 free pages within the CMS storage pool.

Context

- “DEVICE and LINK Statement — OSD Devices” on page 80
- *TCP/IP User’s Guide* for the NETSTAT command

GATEWAY Statement

Use the GATEWAY statement to add static routes to the IP route table and to identify interfaces that are to be used in conjunction with IP multicast support.

The IP route table can be modified by replacing it using the OBEYFILE command, by adding new routes using the RouteD server, or by accepting incoming ICMP redirect packets sent from adjacent machines.

If RouteD is running, static routes defined by the GATEWAY statement are not managed or updated by RouteD. To have RouteD manage these routes, remove them from the GATEWAY statement — RouteD will find out about them dynamically. A simple way to flush the static routes from the IP routing table is to include a GATEWAY statement with no routing information in an OBEYFILE file. See “Chapter 7. Configuring the ROUTED Server” on page 155 for further explanation of using the GATEWAY statement with RouteD.

The first GATEWAY statement of each configuration file processed replaces the existing routing table with the new gateway information. Subsequent GATEWAY statements in the same file add entries to the routing table.

```
                ┌──────────────────────────────────────────────────────────┐
                ▼                                                          │
►►──GATEWAY──────┬─network────┬─first_hop───link_name──┤ Options ├────────┴──────────►◄
                 ├─host───────┤
                 └─DEFAULTNET─┘

Options:

├──┬─max_packet_size─┬──┬─subnet_mask subnet_value─┬──────────────────────────────┤
   └─DEFAULTSIZE─────┘  ├─HOST─────────────────────┤
                        └─0────────────────────────┘
```

Operands

network
: The IP network address in dotted-decimal form. Trailing zero octets may be omitted.

- An example of a class A network is 9.0.0.0 or 9
- An example of a class B network is 129.34.0.0 or 129.34
- An example of a class C network is 192.9.100.0 or 192.9.100

Use the *subnet_mask* and *subnet_value* fields to define a subnetted network.

DEFAULTNET
Specifies the default to use for any network that is not explicitly routed.

You must specify a GATEWAY statement that defines the route to the *first_hop* for the DEFAULTNET keyword to take effect.

host
The host address, specified in dotted-decimal form (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 valid for DEFAULTNET.
- The IP address of a gateway or router that you can reach directly and that forwards messages to the destination network or host.

link_name
The name of the link through which packets are sent to the specified network. The link name must be defined in a LINK statement and cannot be the name of a virtual link.

If the network or host can be reached through an Offload host, then specify the name of the appropriate API link (not the name of the Offload host IP link).

max_packet_size
The maximum transmission unit (MTU) in bytes for the network or host.

This value can be up to 65,535 but cannot be larger than the value specified on the LARGEENVELOPEPOOLSIZE statement.

The special entry DEFAULTSIZE in this field requests that TCP/IP supply a value of 576. IBM recommends you use the following sizes instead of DEFAULTSIZE as the packet size for these networks:

| 1492 bytes for Ethernet 802.3
| 1500 bytes for Fast Ethernet
| 8902 bytes for Gigabit Ethernet
1500 bytes for Ethernet Version 2 IEEE
1500, 2048, or 4096 bytes for IBM Token-Ring
2048, 4352, or 4096 bytes for FDDI
4096 bytes for CLAW
65527 bytes for CTC

If any bridge or router does not perform IP-layer fragmentation of packets, you must select an MTU corresponding to the smallest MTU in use by that bridge or router. Selecting an MTU size that is too large may cause client applications to hang.

subnet_mask
A bit mask, expressed in dotted-decimal form, that shows the bits of the *network* and *host* fields that make up the subnet. The usual practice is to make the bits contiguous in the host field. **Unlike the BSDROUTINGPARMS statement, the high-order octets of the subnet mask that define the A, B, or C network number must be zero. For example, a Class C subnet of Class B network 129.5 would be represented as 0.0.*x.y*.**

If the network does not use subnets, specify a *subnet_mask* of 0 and omit the *subnet_value*.

If this is a host entry, specify a *subnet_mask* of HOST and omit the *subnet_value*.

For subnetted and supernetted routes, the subnet mask can be in several forms, depending upon the whether VARSUBNETTING option of the ASSORTEDPARMS statement is specified, as follows:

- No VARSUBNETTING

  The network portion is coded as zero and the host portion shows the bits that designate the subnet. The bits must be contiguous. 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. For an example, see the GATEWAY statements accompanying Figure 7 on page 95.
- VARSUBNETTING

  Variable-length subnet masks may be used in a single network; that is, subnets with the same network number may 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. For an example, see the GATEWAY statements accompanying Figure 9 on page 97.

  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 so that in the network portion, the supernet portion is zero; the remainder is all 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, the supernet 130.200 with supernet mask of 255.252.0.0, the difference of the class B network mask (255.255.0.0) and the supernet mask, yields a mask of 0.3.0.0. Based on the unmasked bits in the network portion, 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.

The topics "Subnetting" and "Simplified IP Datagram Routing Algorithm with Subnets" in the *TCP/IP Diagnosis Guide* illustrate the concept of subnetting and provide an example that describes how a subnet route is resolved. Subnets and subnet masks are also discussed in the IBM Redbook IP *Network Design Guide*, SG24–2580.

subnet_value

Value of the *subnet_mask*. Each subnet should have a unique dotted-decimal representation. A value of 0 identifies the default route for any subnet of this network that is not specifically routed. This is not supported for routes through Offload devices. Omit the *subnet_value* field if the *subnet_mask* field is 0 or HOST.

If the network has one or more subnets, specify a separate entry in the GATEWAY statement for each subnet. The network part of each subnet GATEWAY entry should be identical (that is, contain the IP network address as if the network had no subnets).

Examples

1. This example illustrates a network that does not use variable subnetting.

Figure 7 shows a host, `VM1`, directly connected to class C networks 193.9.200 and 193.1.2. Neither network has subnets. `VM1` is indirectly connected to class B network 128.84, which is subnetted using the first octet of the host number as the subnet number:

- Subnet 128.84.1 is accessible through host 193.9.200.2.
- Subnet 128.84.55 is accessible through host 193.9.200.100.
- All other subnets of 128.84 are accessible through host 193.1.2.2.

All packets destined for a network that has no entry in the routing table should be routed to host 193.1.2.3. All packets for host `jakespc` should be routed through 193.1.2.2.

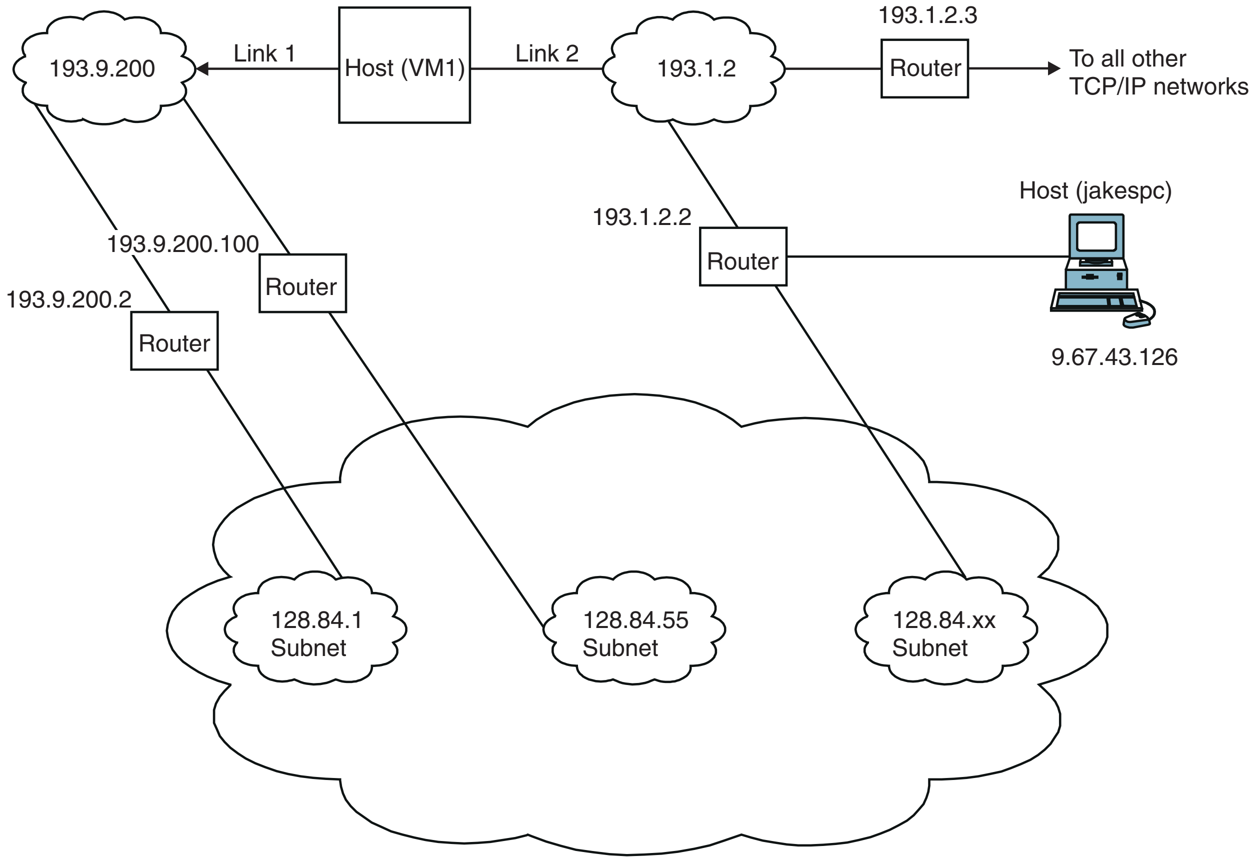


Figure 7. Example of Network Connectivity

An example of the corresponding GATEWAY statement follows.

```
GATEWAY
; IP Address    First Hop     Link     (MTU)      Subnet Mask  Subnet Value
  193.9.200     =             LINK1   DEFAULTSIZE  0
  193.1.2       =             LINK2   DEFAULTSIZE  0
  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        193.1.2.2     LINK2   DEFAULTSIZE  0.0.255.0    0
  DEFAULTNET    193.1.2.3     LINK2   DEFAULTSIZE  0
  9.67.43.126   193.1.2.2     LINK2   DEFAULTSIZE  HOST
```

Note: For the networks not using subnets, the *subnet_mask* is 0 and the *subnet_value* is omitted.

The *subnet_mask* is the same for all instances of the subnet network 128.84. A subnet value of 0 in the *subnet_mask* field identifies the default route for the subnets not explicitly routed.

2. This is another example that shows a GATEWAY statement that is divided by the types of routes used.

```
GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------
;
  130.50        =            TR1       2000         0.0.255.0   0.0.10.0
  193.5.2       =            ETH1      1500         0
  128.240       =            OFFLAN1   1500         0.0.255.0   0.0.5.0
  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.
;
; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------
;
  193.12.2      130.50.10.1  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.
;
; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------
;
  DEFAULTNET    9.67.43.1    FDDI1     DEFAULTSIZE  0
```

Figure 8. Example of route types

3. This is an example of a network that employs variable subnets and supernets.

   Figure 9 on page 97 shows a host, VM1, directly connected to variable subnets 10.2 and 10.2.8. VM1 is indirectly connected to variable subnets 121.2.3.128 and 121.2.4. In addition, VM1 is directly connected to supernet 192.3.200 and indirectly connected to supernet 130.200.

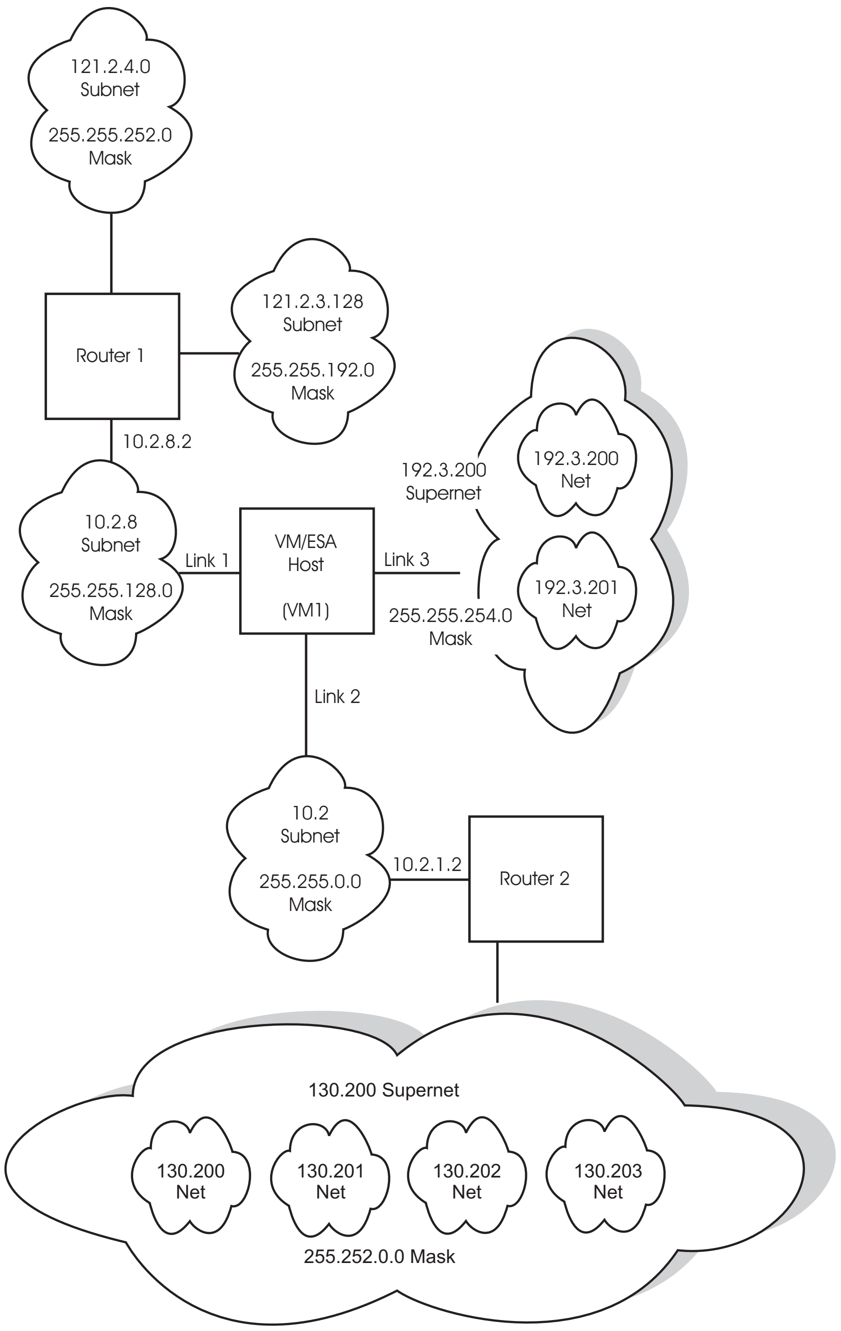


Figure 9. Example of Network Connectivity Using Variable Subnetting

The following is an example of the corresponding GATEWAY statement:

```
GATEWAY
;
; Direct and indirect subnets
;
; (IP) Network  First         Link    Max. Packet  Subnet      Subnet
; Address       Hop           Name    Size (MTU)   Mask        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
;
; (IP) Network  First        Link    Max. Packet  Subnet      Subnet
; Address       Hop          Name    Size (MTU)   Mask        Value
; -----------   ------------  ------- -----------  ----------- --------
;
  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. Subnet 10.2 uses the high-order byte of the host field as the subnet field, and subnet 10.2.8 uses the first two bytes of the host field as the subnet fields.

| Network 121.2 has variable subnets 121.2.3.128 and 121.2.4, and is an example of indirect routing. For the network shown in Figure 9 on page 97, IP datagrams will be delivered to the directly connected router at address 10.2.8.2, which represents the ″first-hop″ toward either of two destination subnets, 121.2.4 or 121.2.3.128.

Networks 130.200 and 192.3.200 are supernets that represent multiple networks; these use the high-order bytes of the network field as the supernet fields. The number of networks is determined by the mask bits in a network field. For example, subnet mask 0.0.1.0 is interpreted as supernet mask 255.255.254.0, while subnet mask 0.3.0.0 is interpreted as supernet mask 255.252.0.0. Each is calculated by taking the difference between the network class mask and the supernet mask. 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 with subnet mask 0.0.1.0, represents networks 192.3.200 and 192.3.201. Supernet 130.200 with subnet mask 0.3.0.0, represents networks 130.200, 130.201, 130.202, and 130.203. Note that the subnet value is zero for the supernets.

Usage Notes

- If the gateway table is empty, the loopback test addresses are still routed properly.
- If a syntax error is found in a GATEWAY statement, the remainder of the statement is ignored. Subsequent GATEWAY statements in the same profile or OBEYFILE are processed.
- Routes are used in the following sequence:
  1. host
  2. subnetwork
  3. network
  4. supernetwork
  5. default
- | An interface can be defined to send multicast datagrams. This can be accomplished by specifying the general multicast group address of 224.0.0.0 for an interface, as in this example:

```
   GATEWAY
; (IP) Network  First         Link      Max. Packet  Subnet      Subnet
; Address       Hop           Name      Size (MTU)   Mask        Value

 224.0.0.0      =             LINK1     DEFAULTSIZE  HOST
```

Sending interfaces can also be defined for specific multicast groups. In the example that follows, the LINK2 link entry identifies that this link is to be used to send datagrams to the 224.1.1.1 multicast address:

```
   GATEWAY
; (IP) Network  First         Link      Max. Packet  Subnet      Subnet
; Address       Hop           Name      Size (MTU)   Mask        Value

 224.1.1.1      =             LINK2     DEFAULTSIZE  HOST
```

The interface used for sending an outbound multicast datagram is selected based on the following precedence:

1. Use the interface associated with a specific socket, as specified with a send(), sendmsg(), or sendto() call for which the MSG_DONTROUTE parameter has also been specified.
2. Use an application-specified interface, as determined through the setsockopt() call for which the IP_MULTICAST_IF parameter has been specified.
3. Use an interface that is associated with a specific multicast group address, perhaps 224.1.1.1
4. Use an interface that is associated with the general multicast group address (244.0.0.0).
5. Use the default network interface (identified by the GATEWAY statement DEFAULTNET operand), provided its associated link is multicast-capable. If this default interface is not multicast-capable, attempts to send multicast datagrams receive an ENETUNREACH error.

- Packet size considerations:
  - Information is transferred over a TCP connection in discrete packets. Each packet includes a TCP header and an IP header. The header size is independent of the amount of user information included, so the larger the packets sent, the less relative bandwidth is consumed by protocol headers. Also, the TCP software layer consumes a fixed amount of CPU time for each packet, independent of the packet size.
  - *max_packet_size* may not exceed the value specified for *lrg_env_size* in the LARGEENVELOPEPOOLSIZE statement. See "LARGEENVELOPEPOOLSIZE Statement" on page 110 for more information about large envelope size. Some networks limit the packet size to a smaller value. For example, while the largest packet size for the Ethernet protocol is 1500 bytes, the largest packet size for the 802.3 protocol is 1492.
  - The actual packet size will be determined by the total network connection:
    - If a locally-attached host has a packet size smaller than yours, transfers to that host will use the smaller size.
    - The Offload processor determines the packet size for all LANs attached to it. Therefore, the GATEWAY *max_packet_size* specification is ignored for Offload processing.
    - The TCP maximum segment size for the IBM 3172 Interconnect Controller Program is 4096. Any specification greater than 4096 will be treated as 4096. 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. To reduce these costs, packets sent through gateways to other networks should use the default size, DEFAULTSIZE, unless all intervening gateways and networks are known to accept larger packets. However, to minimize TCP/IP for VM overhead, larger packets sizes should be used. For additional information, refer to the *z/VM: Performance* book.
– 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). You may use the GATEWAY configuration statement to further reduce packet sizes. For example, the router to network 192.8.4 is reached through router 9.0.0.10., and somewhere along the path, packets larger than 460 bytes are fragmented. Network throughput can be improved, possibly at the expense of additional TCP/IP for VM overhead, using the following GATEWAY statement:

```
GATEWAY
; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------
;
  192.8.4       9.0.0.10     LINK1     460          0
```

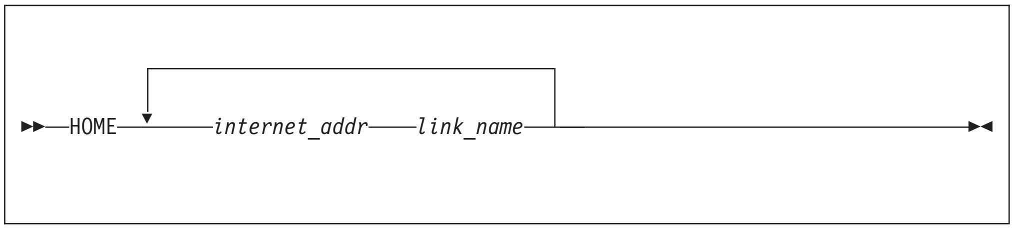
Context

- “ASSORTEDPARMS Statement” on page 50
- “BSDROUTINGPARMS Statement” on page 59
- “DEVICE and LINK Statements” on page 41
- “LARGEENVELOPEPOOLSIZE Statement” on page 110
- “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135
- “Chapter 7. Configuring the ROUTED Server” on page 155

HOME Statement

The HOME statement defines home addresses and their associated link names (called the HOME list).

The first HOME statement of each configuration file processed replaces the existing HOME list with a new one. Subsequent HOME statements in the same file add entries to the list.



Operands

internet_addr
One of the IP addresses valid for this host. The address can be associated with any type of link. It must be specified in dotted-decimal form.

link_name
: The name of the link (defined in a previous LINK statement) associated with the home address.

Examples

This example shows a HOME statement that defines the IP addresses of each link to the host. The corresponding GATEWAY statement is shown in Example 2 on page 96.

```
Home
   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 virtual links; the remaining entries are examples of actual links that are associated with physical IP addresses. Virtual IP addresses are used in outbound IP datagrams. For more information, see "ASSORTEDPARMS Statement" on page 50. If you specify SOURCEVIPA on the ASSORTEDPARMS statement, link VIPA1 provides the virtual IP address for TR1 and ETH1, while link VIPA2 provides it for FDDI1 and SNA1.

Usage Notes

- The home address of an Offload API link is specified on the LINK statement rather than the HOME statement.
- More than one home address can be associated with a link. The first home address specified for a link is its primary one.
- The PRIMARYINTERFACE IP address is used as the source address in the IP header of an outgoing packet if no other source IP address can be found. If the PRIMARYINTERFACE statement is not specified, then the first address in the HOME list is the default local address, which is used for the GETHOSTID function.
- If a TCP application on the VM host system does not specify a local address value, the application receives the default local address, as described in the previous note. This value is set to the first address in the HOME list if a PRIMARYINTERFACE statement has not been associated with a link. For example, if the PRIMARYINTERFACE is set to LINK2, and the HOME list is as follows:

  ```
  HOME
    9.0.28.3      LINK1
    192.6.77.5    LINK2
  ```

  then the default local address would be 192.6.77.5. If no PRIMARYINTERFACE is set, the default local address would be 9.0.28.3.
- When defining virtual IP addresses, observe the following rules and recommendations:
  - Code a primary virtual IP address first in the HOME list or on the PRIMARYINTERFACE statement to serve as the default local address.

    In general, virtual IP addresses can be coded in any order in the HOME list; however, if you specify SOURCEVIPA on the ASSORTEDPARMS statement, the order of addresses is important with respect to how source IP addresses are used for outbound datagrams originating at the host. In this case, TCP/IP behaves as follows:

  - In the HOME list, the virtual IP address that most closely precedes a physical IP address is used as its source IP address.
  - If virtual IP addresses are coded after all physical IP addresses, no virtual addresses are used as source IP addresses.

  Note: See the **Examples** section for more information about configuring the HOME statement when SOURCEVIPA is specified.
- A virtual IP address must be a unique host address in the network and may not duplicate any physical IP address in the network.
- More than one virtual IP address can be defined in one network or subnetwork.
- You can use a virtual IP address as the primary (or only) destination for a z/VM server when the z/VM host is defined for your domain name server. A workstation on the network would use the z/VM server name (translated into the virtual IP address) to access applications on the z/VM server.

• While a virtual IP address can be assigned to each TCP/IP stack in one z/VM system, it is recommended that an internal point-to-point link (for example, virtual CTC) be defined between these stacks. This ensures the virtual IP address used for one z/VM TCP/IP stack that is attached to a failing device (for example, a 3172) can be reached via another z/VM TCP/IP stack that is channel-attached to the same controller through another adapter, or to another controller across the point-to-point link.

• For more information about which routing protocols you can use to achieve non-disruptive TCP-connection fault tolerance, see “Virtual IP Addressing (VIPA)” on page 37.

• If you are using a nameserver to resolve host names via UDP and any of the related resolver configuration files have only one nameserver address coded that specifies a z/VM virtual IP address, the host the nameserver is running on must be configured to use SOURCEVIPA.

| • If you are running MPROUTE or ROUTED, each link should have a unique
| home address.

Examples

This example uses the SOURCEVIPA option for outbound datagrams originating at a z/VM TCP/IP stack.

Select a virtual IP address in the HOME statement as the local address. The address that most closely precedes a physical IP address is used as its local address. 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 virtual IP addresses can be defined to associate a group of interfaces and serve as local addresses. In the previous example, VIPA1 is associated with ETH1 and TR1, and VIPA2 is associated with ETH2 and TR2.

If an outbound datagram is not to contain a source virtual IP address for a particular interface (that is, a physical IP address should always be used), the address and link entries must be suitably ordered, as shown in the following example:

```
HOME
  151.4.1.1     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
```

Table 10 shows how various TCP/IP protocols use a virtual IP address when it is specified as a local address. ″Y″ indicates that the address will be used as the local address, and ″N″ indicates that it will not be used as the local address.

Table 10. Source VIPA Usage Chart

Destination	**ICMP**	**TCP**	**RAW**	**UDP**
Local Interface	N	N	N	N
Local network	Y	Y	Y	Y*
Remote network	Y	Y	Y	Y

Note: * Except for RIP packets

Context

- “ASSORTEDPARMS Statement” on page 50
- “BSDROUTINGPARMS Statement” on page 59
- “DEVICE and LINK Statements” on page 41
- “PRIMARYINTERFACE Statement” on page 119

INFORM Statement

Use the INFORM statement to define a list of users (called the INFORM list) who are to be sent messages in case of serious run-time conditions.

```
            ┌─────────┐
            ▼         │
►►──INFORM────user_id───ENDINFORM──────────────────────────────►◄
```

Operands

user_id
Defines a user to be informed in case of serious run-time error conditions.

Examples

This example shows how to inform users OPERATOR, TCPMAINT, and MAINT of serious errors.

```
Inform
  OPERATOR TCPMAINT MAINT
EndInform
```

Usage Notes

- The ENDINFORM statement ends the INFORM statement. If ENDINFORM is omitted, all subsequent statements will be interpreted as user IDs.
- When a serious error is detected, each user in the current INFORM list is sent a short descriptive message. For example, when the number of objects in any pool displayed by the NETSTAT POOLSIZE command falls below its permit size, each user in the list is notified.
- The first INFORM statement of each configuration file replaces the existing INFORM list. Subsequent INFORM statements in the same file add users to the list.

Context

- “Free Pool Statements” on page 44
- *TCP/IP User’s Guide* for the NETSTAT command

INTERNALCLIENTPARMS Statement

The INTERNALCLIENTPARMS statement is used to configure the Telnet server, an internal client of the TCPIP virtual machine.

```
                          ┌────────────────────────────────┐
                          ▼                                │
►►──INTERNALCLIENTPARMS───┬────────────────────────────┬───┴──────────────►
                          ├─ASYNCHRONOUSINPUT──────────┤
                          │ ┌─CCSTERMNAME TCPIP──┐     │
                          ├─┤                  ├─────┤
                          │ └─CCSTERMNAME──ccccc─┘     │
                          ├─CONNECTEXIT──filename──────┤
                          │ ┌─EOJTIMEOUT 120──┐        │
                          ├─┤               ├────────┤
                          │ └─EOJTIMEOUT──sec─┘        │
                          ├─IGNOREEAUDATA──────────────┤
                          │ ┌─INACTIVE 0──────┐        │
                          ├─┤               ├────────┤
                          │ └─INACTIVE──sec───┘        │
                          │ ┌─LDEVRANGE 0 0FFF─────┐   │
                          ├─┤                    ├───┤
                          │ └─LDEVRANGE──low high──┘   │
                          ├─NOTN3270E──────────────────┤
                          │ ┌─PORT 23──────────────┐   │
                          ├─┤                    ├───┤
                          │ │   ┌────────────┐   │   │
                          │ │   ▼            │   │   │
                          │ └───PORT ──num───┴───┘   │
                          │ ┌─SCANINTERVAL 120──┐      │
                          ├─┤                 ├──────┤
                          │ └─SCANINTERVAL──sec─┘      │
                          │ ┌─TIMEMARK 600────┐        │
                          ├─┤               ├────────┤
                          │ └─TIMEMARK──sec───┘        │
                          ├─TN3270EEXIT──filename──────┤
                          ├─TN3270ENOSCEXIT────────────┤
                          └─TRANSFORM──────────────────┘

►─ENDINTERNALCLIENTPARMS─────────────────────────────────────────────────►◄
```

Operands

ASYNCHRONOUSINPUT

ASYNCHINPUT

ASYNCINPUT

For Telnet LINEMODE connections, causes the Telnet server to signal an attention interrupt to the associated virtual machine, when input is received from the client and the virtual machine has not issued a read. This usually causes the virtual machine to issue a read, allowing the user input to be presented. If this option is not specified, the Telnet server holds client input until the associated virtual machine issues a read.

CCSTERMNAME *ccccc*
String 1 to 5 characters in length specifying the terminal name prefix for line-mode Telnet sessions.

CONNECTEXIT *filename*
The name of the Telnet session connection exit to be loaded. The exit will be called every time a Telnet session connection is established unless the TN3270ENOSCEXIT parameter has been supplied via the OBEYFILE command or as a parameter on the INTERNALCLIENTPARMS statement in the TCP/IP configuration file.

The CONNECTEXIT can be used to control system access based on the client's IP address and the target port number. The exit can also be used to specify an initial CP command (such as `DIAL VTAM`) to be simulated for a client with a transparent mode session.

The search sequence is:
1. The GLOBAL LOADLIB list,
2. *filename* TEXT on any accessed disk,
3. The GLOBAL TXTLIB list.

The CONNECTEXIT parameter is ignored if it is supplied via the OBEYFILE command. The exit interface is described in the *TCP/IP Programmer's Reference*.

| **EOJTIMEOUT** *sec*
| Sets the EOJTIMEOUT interval. This parameter is used in conjunction with
| TN3270E printer support, and causes an EOJ header to be sent to a printer if
| no such header is sent within the specified number of seconds. The default is
| 120 seconds (2 minutes).

IGNOREEAUDATA
Causes the Telnet server to ignore any data associated with Erase All Unprotected (EAU) commands in the data stream received from the host. Ordinarily, any such data is forwarded to the client. Some Telnet clients enforce the restriction that there can be no data associated with an EAU command and require this option in order to function properly.

INACTIVE *sec*
Sets the Telnet 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, which prevents inactivity checking. This number is an integer from 0 to 99,999,999.

LDEVRANGE *low high*
Hexadecimal logical device number range between **0** and **FFFF** to be used for incoming Telnet connections. Do not set the end of the range larger than the maximum logical device number defined by the CP SET MAXLDEV command. Since logical device numbers are unique within the VM system, there is no guarantee that other service machines will not use the same device range that is assigned to TCP/IP.

If LDEVRANGE is not specified, logical device numbers in the range from **0** to **0FFF** will be used.

NOTN3270E
Prevents the Telnet server from negotiating sessions based on the TN3270E protocol. Some Telnet clients might not handle TN3270E negotiation correctly, in which case this parameter can be used to allow them to function correctly. However, Telnet-based printer sessions are not supported if this parameter is specified.

PORT *num*
Accepts incoming Telnet requests on a specified port number rather than the default port 23. This parameter may be specified multiple times to accept incoming Telnet requests on any of several different ports. The port numbers specified should have corresponding PORT statements that reserve them for the special user identifier INTCLIEN, which represents the Telnet server.

The PORT parameter is ignored if it is supplied via the OBEYFILE command.

SCANINTERVAL *sec*
Sets the interval between scanning for idle connections or connections waiting to receive a TIMEMARK. If the value is specified as 0, or if the parameter 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.

TIMEMARK *sec*
Sets the timemark time-out to a specified number of seconds. The Telnet server sends a TIMEMARK option to each connection at this interval. If there is any reply, it is ignored. If TCP cannot send the option within its connection time-out interval, it informs the Telnet server. The Telnet server closes the connection. The default is 600 seconds (10 minutes).

TN3270EEXIT *filename*
The name of the Telnet printer management exit. The exit is called every time a Telnet printer connection is established or terminated.

The TN3270EEXIT can be used to control system access based on the client's IP address and port number, the local port number, the logical unit name associated with the session by the client, and the user identifier and virtual device address associated with the session through the TN3270E statement (see "TN3270E Statement" on page 129).

The search sequence is:
1. The GLOBAL LOADLIB list,
2. *filename* TEXT on any accessed disk,
3. The GLOBAL TXTLIB list.

The TN3270EEXIT parameter is ignored if it is supplied via the OBEYFILE command. The exit interface is described in the *TCP/IP Programmer's Reference*.

TN3270ENOSCEXIT
Prevents the Telnet server from calling the session connection exit for telnet printer sessions only. Otherwise, the session connection exit will be called, if it is loaded, for all telnet sessions.

TRANSFORM
Causes the Telnet server to load a 3270 transform program. File TNSIMHPI TEXT must be accessible by the server, and additional virtual storage might be needed. This file is available only with third-party products; it is not supplied with TCP/IP for VM. The TRANSFORM parameter is ignored if it is supplied via the OBEYFILE command.

Examples

This example shows that TNEXIT1 TEXT is to be loaded and used as the Telnet session connection exit.

```
InternalClientParms
   ConnectExit TNEXIT1
EndInternalClientParms
```

Usage Notes

- If a parameter name is misspelled or if the value specified is not valid, the parameter is ignored and the default is used.
- When using the OBEYFILE command to modify the INTERNALCLIENTPARMS statement, keep these rules in mind:
  - The values of all parameters except CONNECTEXIT, PORT, TN3270EEXIT, and TRANSFORM may be changed.
  - An INTERNALCLIENTPARMS parameter that may be changed but is not specified assumes its default value or setting.

Context

- "PORT Statement" on page 117
- "TN3270E Statement" on page 129

IPROUTEPOOLSIZE Statement

Use the IPROUTEPOOLSIZE statement to set the initial number of IP route control blocks. IP route control blocks are used to hold information about IP routes. Each one requires 128 bytes.

```
    ┌─IPROUTEPOOLSIZE 600───────┐
►►──┼───────────────────────────┼──────────────────────────────────────────────►◄
    └─IPROUTEPOOLSIZE number────┘
```

Operands

number
: The initial number of IP route control blocks in the free pool.

Examples

This example shows an IPROUTEPOOLSIZE statement that defines the number of IP route control block to be the default of 600.

```
IProutePoolSize   600
```

Usage Notes

- Each entry in the IP routing table, whether created using the GATEWAY statement, by Internet Control Message Protocol (ICMP) Redirect, or by RouteD, requires one IP route control block. When an entry is deleted from the IP routing table, the corresponding IP route control block is not immediately returned to the pool, if the IP-down route cache or a TCP connection refers to it. The control block is returned to the pool the next time TCP/IP tries to use the saved reference and finds that the entry is no longer valid.
- The system will attempt to dynamically allocate 10% more IP route control blocks any time the IP route control block free pool becomes empty. You can use the NETSTAT POOLSIZE command to monitor how many IP route control blocks your system is using. To avoid dynamic allocation of IP route control blocks during operation, use the IPROUTEPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- “GATEWAY Statement” on page 92
- “Chapter 7. Configuring the ROUTED Server” on page 155
- *TCP/IP User’s Guide* for the NETSTAT command

KEEPALIVEOPTIONS Statement

Use the KEEPALIVEOPTIONS statement to specify the operating parameters of the TCP/IP keep-alive mechanism. The parameters apply to all TCP connections for which keep-alive has been activated, through either the setsockopt() call of the C socket interface or the TcpOption call of the Pascal interface.

```
                    ┌─INTERVAL 120─────┐   ┌─SENDGARBAGE FALSE───────┐
►►──KEEPALIVEOPTIONS─┼──────────────────┼───┼─────────────────────────┼──────────►
                    └─INTERVAL minutes─┘   └─SENDGARBAGE─┬─TRUE──┬──┘
                                                         └─FALSE─┘

►─ENDKEEPALIVEOPTIONS─────────────────────────────────────────────────────────►◄
```

Operands

INTERVAL *minutes*
The number of minutes TCP/IP waits after last receiving a packet for a TCP connection before it sends a keep-alive packet for it. The default is 120 minutes (2 hours).

SENDGARBAGE
Specifies whether the keep-alive packets sent by TCP/IP contain one byte of random data.

FALSE
The keep-alive packet will not contain random data. This is the default.

TRUE
The keep-alive packet will contain one byte of random data and an invalid sequence number, assuring that the data is not accepted by the remote TCP/IP.

Usage Notes

- Some hosts cannot properly respond to keep-alive packets containing no data. If your network includes such hosts, set the SENDGARBAGE parameter to TRUE.
- The ENDKEEPALIVEOPTIONS statement specifies the end of the KEEPALIVEOPTIONS information. If it is omitted, subsequent entries will generate error messages.

LARGEENVELOPEPOOLSIZE Statement

Use the LARGEENVELOPEPOOLSIZE statement to set the initial number and size of large envelopes. Large envelopes are used to hold UDP datagrams larger than 2,048 bytes while they are being sent and while they are waiting for an application program to receive them. They are also used to hold IP datagram fragments during reassembly.

A large envelope is used only if a packet does not fit into a small envelope.

```
     ┌─LARGEENVELOPEPOOLSIZE 50 8192──────────┐
►►───┼────────────────────────────────────────┼──────────────────────────────►◄
     │                               ┌─8192─┐ │
     └─LARGEENVELOPEPOOLSIZE number──┼──────┼─┘
                                     └─size─┘
```

Operands

number
: The initial number of large envelopes in the free pool. The default is 50.

size
: The size of each large envelope (in bytes); this *size* determines the maximum number of bytes that can be held by a large envelope. The default is 8192. Only the following values or their alternatives (shown in parenthesis) may be specified:

  512
  1024 (1K)
  2048 (2K)
  4096 (4K)
  8192 (8K)
  16384 (16K)
  32768 (32K)
  65535 (64K)

Examples

This example shows a LARGEENVELOPEPOOLSIZE statement that defines the default number and size of large envelopes.

```
LargeEnvelopePoolSize   50   8192
```

Usage Notes

- A large envelope size of 8192, which is the default, actually accommodates packets up to 9,216 bytes long.
- If any *max_packet_size* on the GATEWAY statement is greater than 2,048, then additional large envelopes will be needed for applications such as FTP.
- If *size* is not specified, large envelopes can hold up to 9,216 bytes of data, so packets up to 9,216 bytes can be sent and received.
- Specify *size* parameter to establish the size of the largest packet that can be sent and received.

Note: The specified *size* value must equal or exceed that specified for regular-sized envelopes (defined by the ENVELOPEPOOLSIZE statement). Failure to do so will result in TCP/IP stack initialization errors.

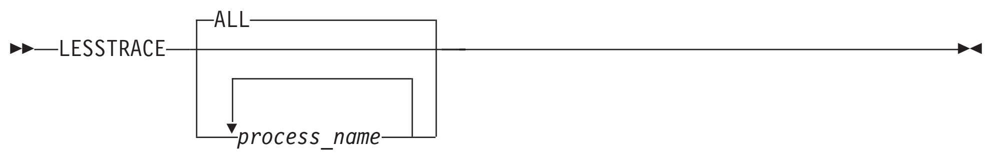
- Matching *size* values should be specified for each of the two hosts for which a CTC connection is defined.
- The system will attempt to dynamically allocate 10% more large envelopes any time the large envelope free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many large envelopes your system is using. To avoid dynamic allocation of large envelopes during operation, use the LARGEENVELOPEPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- “DATABUFFERPOOLSIZE Statement” on page 63
- “ENVELOPEPOOLSIZE Statement” on page 89
- “GATEWAY Statement” on page 92
- *TCP/IP User’s Guide* for the NETSTAT command

LESSTRACE Statement

Use the LESSTRACE statement to turn off detailed run-time tracing for the specified internal TCP/IP processes.



Operands

process_name
: Detailed tracing information is suppressed for the named process. Refer to Table 11 on page 131 for valid process names.

Usage Notes

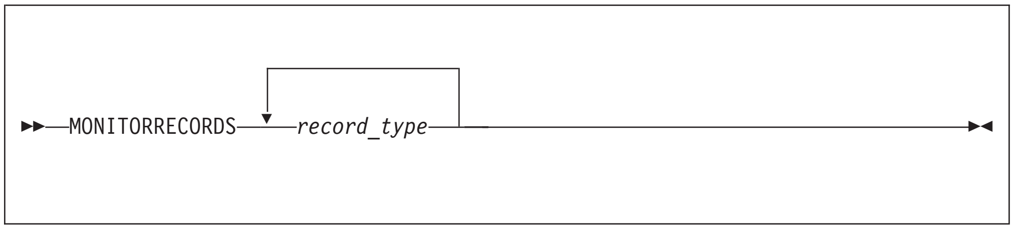
- If no process names are specified, detailed tracing is suppressed for all processes.
- LESSTRACE affects only detailed tracing. To turn off basic tracing, use NOTRACE.

Context

- “TRACE Statement” on page 130
- “MORETRACE Statement” on page 113
- “NOTRACE Statement” on page 114
- “TRACEONLY Statement” on page 131

MONITORRECORDS Statement

Use the MONITORRECORDS statement to select the monitor data records that are produced by TCPIP. This statement is ignored if it is supplied through the OBEYFILE command interface.



Operands

record_type
Specifies the name of a monitor data record type and is one of the following:

Type	**Records produced**
ALLRECORDS	all
CPU	CPU consumption
CLIENTS	client activity
HOMES	home address configuration
LINKS	link configuration and activity
MIB	management information base data
MOSTRECORDS	all except Scheduler (default)
POOLS	storage pool configuration and use
SCHEDULE	Scheduler activity
TCP-SESSIONS	TCP session activity
TREES	hash tree size
UDP-SESSIONS	UDP session activity

Usage Notes

- The MONITORRECORDS statement determines which performance monitor data records are produced, if any. The list of record names is terminated by end of input or by another configuration statement. An empty list, that is, just the MONITORRECORDS statement alone, enables production of the monitor records in the MOSTRECORDS record type alias.
- In order for this statement to take effect, the TCPIP virtual machine must be authorized to create monitor data records by having an OPTION APPLMON statement in its User Directory entry.
- To collect monitor data, the APPLDATA class must be enabled for both SAMPLE and EVENT recording for the TCPIP virtual machine and a monitor writer must be active. For example, these CP commands would cause monitor data produced by the virtual machine named TCPIP to be collected.

```
MONITOR SAMPLE ENABLE APPLDATA USER TCPIP
MONITOR EVENT  ENABLE APPLDATA USER TCPIP
```

These CP commands would cause any virtual machine's monitor data to be collected.

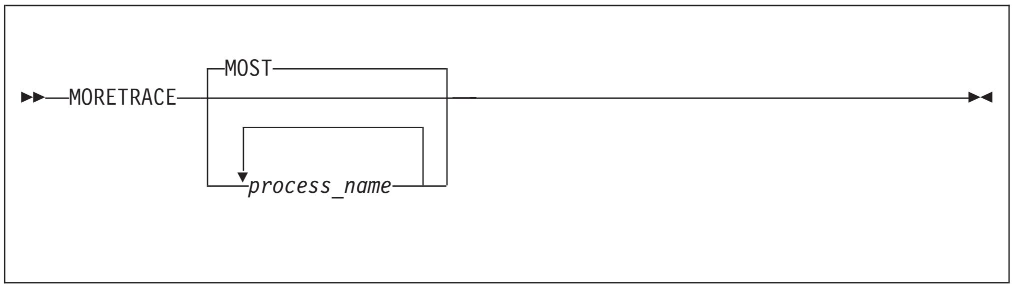
```
MONITOR SAMPLE ENABLE APPLDATA ALL
MONITOR EVENT  ENABLE APPLDATA ALL
```

- To ensure that TCPIP configuration information is included in the monitor data file, start the monitor and a monitor writer before initializing the TCPIP virtual machine.

For more information about performance monitoring, see the *z/VM: Performance* book.

MORETRACE Statement

Use the MORETRACE statement to turn on detailed run-time tracing for the specified internal TCP/IP processes.



Operands

process_name
: Detailed tracing information is enabled for the named process. Refer to Table 11 on page 131 for valid process names.

Usage Notes

- If no process names are specified, detailed tracing is enabled for the processes in the MOST process name alias.
- If a TRACEONLY statement is used, the trace is restricted to the processes that are related to the selected users, devices, or IP addresses.

Context

- "TRACE Statement" on page 130
- "TRACEONLY Statement" on page 131
- "LESSTRACE Statement" on page 111
- "NOTRACE Statement" on page 114

NOSCREEN Statement

Use the NOSCREEN statement to divert run-time trace output from the TCPIP virtual machine console back to the most recently used TRACE file.

```
►►──NOSCREEN──────────────────────────────────────────────────────►◄
```

Operands

The NOSCREEN statement has no parameters.

Usage Notes

The NOSCREEN statement diverts trace information currently directed to the TCPIP console back to the most recently used trace file; any data previously contained in the trace file is destroyed. If no trace file was previously identified via a FILE statement, information will be written to the file **DEBUG TRACE**.

Context

- “TRACE Statement” on page 130
- “FILE Statement” on page 90

NOTRACE Statement

Use the NOTRACE statement to turn off basic and detailed run-time tracing for the specified TCP/IP processes.

```
              ┌─ALL────────────────┐
►►──NOTRACE───┼────────────────────┼──────────────────────────────►◄
              │  ┌───────────────┐ │
              │  ▼               │ │
              └──process_name────┴─┘
```

Operands

process_name
: The basic and detailed run-time tracing information is suppressed for the named process. Refer to Table 11 on page 131 for valid process names.

Examples

This example shows a NOTRACE statement that turns off all tracing for the TELNET process.

```
NOTRACE TELNET
```

Usage Notes

- If no process names are specified, basic and detailed tracing is suppressed for all processes.

- Processing of the NOTRACE statement will not cause an open trace file to be closed; a FILE or SCREEN statement must be used to close an open file.

Context

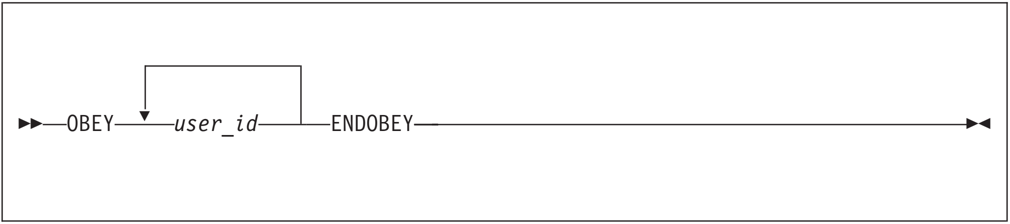
- “TRACE Statement” on page 130
- “LESSTRACE Statement” on page 111
- “MORETRACE Statement” on page 113

OBEY Statement

Use the OBEY statement to define which users may use the following privileged TCP/IP functions:

- The OBEYFILE command
- The NETSTAT CP command
- The NETSTAT DELARP command
- The NETSTAT DROP command
- The NETSTAT RESETPOOL command
- The ROUTED SMSG command
- Any program using rawIP functions
- SMSG functions of the FTP and RouteD servers

The first OBEY statement of each configuration file replaces the existing OBEY list with the new list. Subsequent OBEY statements in the same file add users to the list.



Operands

user_id
A user who is authorized to use privileged TCP/IP functions.

Examples

This example shows an OBEY statement that identifies six users who are to be obeyed.

```
Obey
  OPERATOR TCPMAINT SNMPD SNMPQE MPROUTE ROUTED REXECD DHCPD
EndObey
```

Usage Notes

- The ENDOBEY statement ends the OBEY statement. If ENDOBEY is omitted, all subsequent statements are considered to be entries in the OBEY list.
- TCP/IP servers that must be in the OBEY list in order to function properly are:
  - MPROUTE
  - REXECD
  - ROUTED

- – SNMPD
- – SNMPQE

Context

- “Changing the TCP/IP Configuration with the OBEYFILE Command” on page 135
- “Chapter 7. Configuring the ROUTED Server” on page 155
- *TCP/IP User’s Guide* for the NETSTAT command

PERMIT Statement

Use the PERMIT statement to identify users who are allowed to use TCP/IP services.

The first PERMIT statement of each configuration file replaces the existing PERMIT list with the new list. Subsequent PERMIT statements in the same file add users to the list.

```
              ┌────────────┐
              ▼            │
►►──PERMIT────user_id──────┴──ENDPERMIT──────────────────────────────►◄
```

Operands

user_id
A user who is allowed to use general TCP/IP services.

Usage Notes

- The PERMIT statement is effective only when PERMITTEDUSERSONLY is specified on the ASSORTEDPARMS statement.
- The ENDPERMIT statement ends the PERMIT statement. If ENDPERMIT is omitted, all subsequent statements will be interpreted as user IDs.
- Once a user has been added to the permit list, the user cannot be removed using an obey file. TCP/IP must be restarted.
- Users specified on the AUTOLOG, INFORM, OBEY, or PORT statements do not need to be specified on the PERMIT statement.
- If a user is specified on both the PERMIT and RESTRICT statements, the user will not be able to use TCP/IP services.
- If an unauthorized user attempts to use TCP/IP services, the message

  ```
  date time Unauthorized TCP/IP access attempt by userid
  ```

  will be logged by the TCPIP virtual machine in both its trace file (if active) and to a special authorization audit file named TCPIP AUTHLOG A.

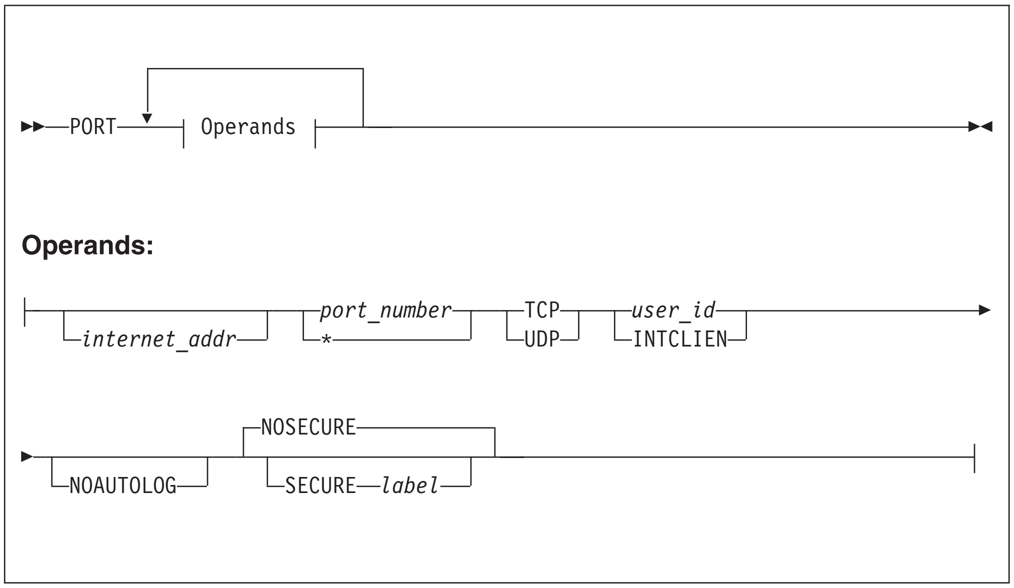
  You may override the file specification by issuing a FILEDEF command for AUTHLOG in the postlude subroutine of the TCPIPXIT EXEC. It must be defined with the DISP MOD option.

Context

- "ASSORTEDPARMS Statement" on page 50
- "RESTRICT Statement" on page 121
- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

PORT Statement

Use the PORT statement to reserve a port for a specific user, to designate a port as secure, or to disable TCP/IP's automatic server restart function.



Operands

internet_addr
: The home address associated with the port to be reserved. If this operand is omitted, the port is reserved for the client regardless of the home address associated with a connection. Otherwise, the port is reserved for the client only for connections associated with the specific home address.

port_number
: The port number to be reserved.

  If you specify an asterisk (*) instead of a port number, all ports to which *user_id* is permitted to bind are treated as secure. However, no connections can be made from *user_id* when this is done.

TCP

UDP
: The protocol that will be used on the reserved port.

user_id
: The virtual machine that may use this port/protocol combination.

  The special value INTCLIEN assigns the port to the internal Telnet server rather than to a client virtual machine.

NOAUTOLOG
: Indicates that TCP/IP is not to restart *user_id* if the user stops listening to this

port. Use NOAUTOLOG when the user is in the AUTOLOG list and server availability is to be under manual control. Please note that NOAUTOLOG will not prevent a restart of the client stemming from circumstances beyond the termination of the listening connection.

NOSECURE
Indicates that the port is not to be considered secure. This is the default.

SECURE
Indicates that the port is secure and that any connections accepted for it will be handled according to the Secure Socket Layer (SSL) protocol.

label
Identifies the server certificate to be used for connections to the secure port.

Examples

This example shows four servers that provide World-Wide Web services, but one, HTTPD4, is not to be monitored by TCP/IP. No other user may establish a TCP connection on port 80.

```
Autolog
  HTTPD1 0
  HTTPD2 0
  HTTPD3 0
  HTTPD4 0
EndAutolog

Port
  80 TCP HTTPD1
  80 TCP HTTPD2
  80 TCP HTTPD3
  80 TCP HTTPD4 NOAUTOLOG
```

The following example shows a group of secure ports:

```
Port
   21 TCP FTPSRV15 SECURE CERT512            ; FTP SERVER
    * TCP FTPSRV15 SECURE CERT512            ; FTP SERVER
```

In this example:

- Port 21, used by the FTPSRV15 server, is secure, and CERT512 is the certificate to be used for connections to this port.
- All ports that the FTPSRV15 server binds to will be considered secure.

Usage Notes

- To prevent the use of well-known ports (0 through 1023) by users that do not have a reserved port, specify the RESTRICTLOWPORTS option on ASSORTEDPARMS.
- A user who is assigned a port is considered to be in the PERMIT list.
- A user in the OBEY list may use any port, without restriction.
- The PORT statement may appear more than once.
- To remove the reservation for a port, the PORT statement must be deleted from the configuration file and TCP/IP must be restarted.
- The PORT statement for the internal Telnet server must specify the TCP protocol and user INTCLIEN. The port number must match the value(s) specified on the INTERNALCLIENTPARMS statement. Using the defaults, the PORT statement for the Telnet server would be

```
PORT
    23 TCP INTCLIEN
```

The Telnet server automatically starts if a TCP port is reserved for INTCLIEN. To disable the Telnet server, omit or comment out this PORT statement.

- The NOAUTOLOG operand is ignored if an asterisk (*) is specified for the port number.
- If you are using a secure FTP client (as shown in the example above), you must specify an asterisk (*) for the port number. This allows any port that the client uses for the data connection to be treated as a secure port. The FTP client must use passive mode.

Context

- “ASSORTEDPARMS Statement” on page 50
- “AUTOLOG Statement” on page 57
- “INTERNALCLIENTPARMS Statement” on page 105
- “PERMIT Statement” on page 116

PRIMARYINTERFACE Statement

Use the PRIMARYINTERFACE statement to specify which link’s home address to use as the default local address.

The primary interface is the address that is inserted as the source internet address in an IP header, when communicating to a destination through an indirect route.

Only one link can be assigned as the primary interface. If multiple PRIMARYINTERFACE statements are specified, the last statement is used.

Except when the SOURCEVIPA option of the ASSORTEDPARMS statement is used, the PRIMARYINTERFACE statement does not affect on outbound traffic to a network when multiple network interfaces are in use. For more information on how outbound traffic is routed to a network in the presence of multiple network interfaces, see “Multiple Interface Network Support” on page 37.

```
►►──PRIMARYINTERFACE link_name──────────────────────────────────────────►◄
```

Operands

link_name
: The name of a link (as defined in a LINK statement) that is to be the primary interface.

Examples

This example shows a PRIMARYINTERFACE statement that specifies a token-ring device:

```
DEVICE OSALCS1 LCS 3400 netman
LINK TR1 IBMTR 0 OSALCS1

DEVICE GDLVMWEB CTC F211
```

```
LINK WEB CTC 1 VMWEB

DEVICE CTC2  CTC  0502
LINK BJACK   CTC 1 CTC2

HOME
    9.227.55.27  BJACK
    9.227.44.100  TR1
    9.227.55.9   WEB

PRIMARYINTERFACE  TR1
```

You can verify the HOME entry is the primary by using the NETSTAT HOME command. The first entry in the list will be used as the primary.

```
Home address list:

Address           Link
-------           ------
9.227.44.100      TR1
9.227.55.27       BJACK
9.227.55.9        WEB
Ready;
```

Usage Notes

- Because you do not include Offload interfaces in the HOME list, you must include one of them on the PRIMARYINTERFACE statement.
- If you are using the PRIMARYINTERFACE statement, it must appear after the HOME statement.
- The primary interface will always appear first in the list of host addresses displayed by NETSTAT HOME.
- If the PRIMARYINTERFACE statement is not specified, then the first address in the HOME list is the default local address.

Context

- “DEVICE and LINK Statements” on page 41
- “HOME Statement” on page 100

RCBPOOLSIZE Statement

Use the RCBPOOLSIZE statement to set the initial number of raw IP control blocks (RCBs). RCBs are used to hold information about internet protocols opened by a client program, either through a RawIPOpen Pascal interface call, or a socket() call with TYPE=SOCK_RAW. Each one requires 228 bytes.

```
   ┌─RCBPOOLSIZE 50─────┐
►►─┼────────────────────┼───────────────────────────────────────────►◄
   └─RCBPOOLSIZE number─┘
```

Operands

number
The initial number of RCBs in the free pool. The default is 50.

Examples

This example shows an RCBPOOLSIZE statement specifying the default number of RCBs.

```
RCBpoolSize   50
```

Usage Notes

- Each RawIPOpen or socket() call requires one RCB, which is freed by the corresponding RawIPClose or close() call.
- The system will attempt to dynamically allocate 10% more RCBs any time the RCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many RCBs your system is using. To avoid dynamic allocation of RCBs during operation, use the RCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

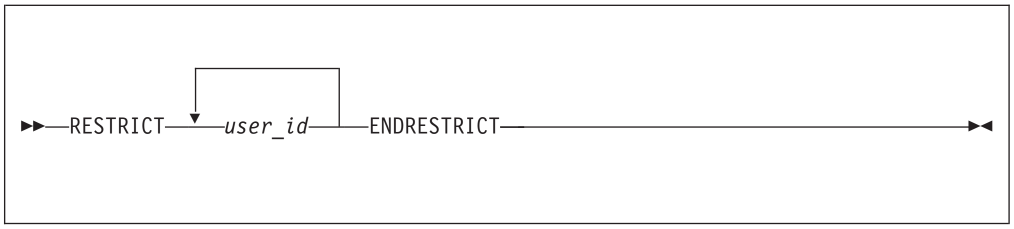
Context

- *TCP/IP User's Guide* for the NETSTAT command

RESTRICT Statement

Use the RESTRICT statement to identify users who are prohibited from using TCP/IP services.

The first RESTRICT statement of each configuration file processed replaces the existing RESTRICT list with the new one. Subsequent RESTRICT statements in the same file add users to the list.



Operands

user_id
: A user or group of users that are prohibited from using TCP/IP services.

An asterisk (wild-card character) can be used at the end of a user ID to match any character. For example, NONIBM* matches all user IDs starting with NONIBM, such as NONIBM, NONIBM1, NONIBMA, and NONIBMYZ.

Examples

This example shows how to deny TCP/IP services to user BADGUY and to any user ID beginning with the characters "VEND":

```
Restrict
  VEND*
  BADGUY
EndRestrict
```

Usage Notes

- The ENDRESTRICT statement ends the RESTRICT statement. If ENDRESTICT is omitted, all subsequent statements are interpreted as user IDs.

- If a user is specified on both the PERMIT and RESTRICT statements, the user will not be able to use TCP/IP services.
- If a restricted user attempts to use TCP/IP services, the message

  ```
  date time Unauthorized TCP/IP access attempt by userid
  ```

  will be logged by the TCPIP virtual machine in both its trace file (if active) and to a special authorization audit file named TCPIP AUTHLOG A. You may override the file specification by issuing a FILEDEF command for AUTHLOG in the postlude subroutine of the TCPIPXIT EXEC. The file must be defined with the DISP MOD option.

Context

- "PERMIT Statement" on page 116
- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

SCBPOOLSIZE Statement

Use the SCBPOOLSIZE statement to set the initial number of socket control blocks (SCBs). SCBs are used to hold information about TCP connections and UDP ports. Each one requires 228 bytes.

```
   ┌─SCBPOOLSIZE 256─────┐
►►─┤                     ├──────────────────────────────────────────►◄
   └─SCBPOOLSIZE number──┘
```

Operands

number
: The initial number of SCBs in the free pool. The default is 256.

Examples

This example shows an SCBPOOLSIZE statement specifying the default number of SCBs.

```
SCBpoolSize   256
```

Usage Notes

- The system will attempt to dynamically allocate 10% more SCBs any time the SCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many SCBs your system is using. To avoid dynamic allocation of SCBs during operation, use the SCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- Running out of SCBs prevents the opening of new TCP connections and UDP ports.

Context

- *TCP/IP User's Guide* for the NETSTAT command

SCREEN Statement

Use the SCREEN statement to direct TCP/IP stack run-time trace output to the console and close any current trace file.

```
►►──SCREEN──────────────────────────────────────────────────────────────►◄
```

Operands

None

Context

- “TRACE Statement” on page 130
- “NOSCREEN Statement” on page 114

SKCBPOOLSIZE Statement

Use the SKCBPOOLSIZE statement to set the initial number of socket interface control blocks (SKCBs). SKCBs are used to hold information about communication endpoints established with the socket(), accept(), or takesocket() calls. Each one requires 785 bytes.

```
     ┌─SKCBPOOLSIZE 256──────┐
►►───┼───────────────────────┼──────────────────────────────────────────►◄
     └─SKCBPOOLSIZE number───┘
```

Operands

number
: The initial number of SKCBs in the free pool. The default is 256.

Examples

This example shows an SKCBPOOLSIZE statement specifying the default number of SKCBs.

```
SKCBpoolSize   256
```

Usage Notes

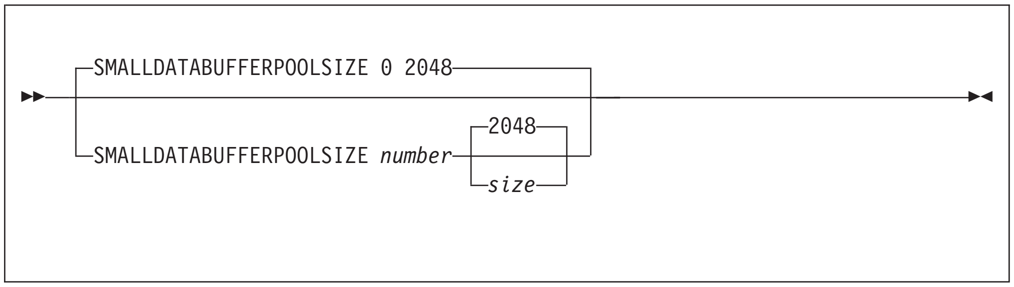
- Each communication endpoint requires one SKCB, which is freed by a close() call.
- The system will attempt to dynamically allocate 10% more SKCBs any time the SKCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many SKCBs your system is using. To avoid dynamic allocation of SKCBs during operation, use the SKCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- *TCP/IP User's Guide* for the NETSTAT command

SMALLDATABUFFERPOOLSIZE Statement

Use the SMALLDATABUFFERPOOLSIZE statement to set the initial number and size of small data buffers. Small data buffers usually hold 2,048 bytes of data, but may be configured to hold up to 4,096 bytes of data.


SMALLDATABUFFERPOOLSIZE 0 2048
SMALLDATABUFFERPOOLSIZE *number*
2048
size

Operands

number
: The initial number of data buffers in the small data buffer pool. The default is 0 — no initial small data buffer pool.

size
: The size of each small data buffer (in bytes). The default size is 2048. Only one of the following values or their alternatives (shown in parenthesis) can be specified:

  2048 (2K)

  3072 (3K)

  4096 (4K)

Examples

This example shows a SMALLDATABUFFERPOOLSIZE statement specifying the number of small data buffers to be 1200.

```
SmallDataBufferPoolSize   1200
```

Usage Notes

- The system will attempt to dynamically allocate 10% more small data buffers any time the small data buffer free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many small data buffers your system is using. To avoid dynamic allocation of small data buffers during operation, use the SMALLDATABUFFERPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- "DATABUFFERPOOLSIZE Statement" on page 63
- "TINYDATABUFFERPOOLSIZE Statement" on page 128
- *TCP/IP User's Guide* for the NETSTAT command

START Statement

Use the START statement to start a device that is currently stopped. This statement is usually specified at the end of the configuration file.

```
►►──START──device_name────────────────────────────────────────────────────►◄
```

Operands

device_name
: The name of the device to start. This must be the same *device_name* specified in a DEVICE statement.

Examples

This example shows START statements that start LCS, Offload, and SNALINK devices.

```
Start LCS1
Start LCS2
Start OFF1
Start SNALU0
```

Usage Notes

- Each device to be started requires a separate START statement.
- The START statement can also be used in an obey file to start:
  - A newly-defined device;
  - A device stopped with the STOP statement;
  - A device that failed to initialize when TCP/IP started.

Context

- "DEVICE and LINK Statements" on page 41
- "STOP Statement"
- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

STOP Statement

Use the STOP statement in an obey file to stop a device that is currently started.

```
►►──STOP──device_name─────────────────────────────────────────────────────►◄
```

Operands

device_name
: The name of the device to be stopped. This must be the same *device_name* specified in a DEVICE statement.

Usage Notes

- Storage used by the device driver is not freed; it is reused the next time the device is started.
- Each device to be stopped requires a separate STOP statement.

Context

- "DEVICE and LINK Statements" on page 41
- "START Statement" on page 125
- "Changing the TCP/IP Configuration with the OBEYFILE Command" on page 135

SYSCONTACT Statement

Use the SYSCONTACT statement to specify the value of the SNMP object *sysContact*. SYSCONTACT is the textual identification of the contact person for this host, together with information about how to contact them.

```
►►──SYSCONTACT──contact_info──ENDSYSCONTACT──────────────────────────────►◄
```

Operands

contact_info
Up to 254 characters of name and contact information for the administrator of this host. TCP/IP converts the specified text to upper case before storing it.

Examples

This example shows a SYSCONTACT statement that defines the contact person and phone number responsible for this host:

```
SysContact
   Main Operator (555-1234)
EndSysContact
```

Usage Notes

The ENDSYSCONTACT statement specifies the end of the SYSCONTACT information.

Context

- "Chapter 22. Configuring the SNMP Servers" on page 513

SYSLOCATION Statement

Use the SYSLOCATION statement to specify the value of the SNMP object *sysLocation*. SYSLOCATION is the physical location of this host (for example, *telephone closet, 3rd floor*).

```
►►──SYSLOCATION──location_info──ENDSYSLOCATION──────────────────────────►◄
```

Operands

location_info
: Up to 254 characters describing the physical location of this node. TCP/IP converts the specified text to upper case before storing it.

Examples

This example shows a SYSLOCATION statement that defines the physical location of the host:

```
SysLocation
   First floor Computer Room
EndSysLocation
```

Usage Notes

The ENDSYSLOCATION statement specifies the end of the SYSLOCATION information.

Context

- "Chapter 22. Configuring the SNMP Servers" on page 513

TCBPOOLSIZE Statement

Use the TCBPOOLSIZE statement to set the initial number of TCP control blocks (TCBs). TCBs are used to hold information about TCP connections. Each one requires 624 bytes.

```
      ┌─TCBPOOLSIZE 256──────┐
►►────┤                      ├──────────────────────────────────────────►◄
      └─TCBPOOLSIZE number───┘
```

Operands

|

number
: The initial number of TCBs in the free pool. The default is 256; the minimum is 6.

Examples

This example shows a TCBPOOLSIZE statement setting the number of TCBs to the default of 256.

```
TCBpoolSize   256
```

Usage Notes

- The system will attempt to dynamically allocate 10% more TCBs any time the TCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many TCBs your system is using. To avoid dynamic allocation of TCBs during operation, use the TCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- *TCP/IP User's Guide* for the NETSTAT command

TIMESTAMP Statement

Use the TIMESTAMP statement to select when to write time stamps in the trace and console files, in order to show when events happened.

```
   ┌─TIMESTAMP PREFIX──────────┐
►►─┼───────────────────────────┼──────────────────────────────────────►◄
   └─TIMESTAMP──┬─num_msg─┬────┘
                └─PREFIX──┘
```

Operands

num_msg
: A number that indicates when time stamps are displayed with a message. If 0 is specified, no time stamps are displayed. Otherwise, time stamps are generated every *num_msg* messages.

PREFIX
: Specifies that a time stamp prefixes every message. This is the default.

Examples

This example generates a time stamp each time 5 messages have been displayed.

```
TIMESTAMP 5
```

Context

- "TRACE Statement" on page 130

TINYDATABUFFERPOOLSIZE Statement

Use the TINYDATABUFFERPOOLSIZE statement to set the initial number of tiny data buffers. Tiny data buffers hold 256 bytes of data.

```
   ┌─TINYDATABUFFERPOOLSIZE 0──────┐
►►─┼───────────────────────────────┼──────────────────────────────►◄
   └─TINYDATABUFFERPOOLSIZE number─┘
```

Operands

| *number*
: The initial number of data buffers in the tiny data buffer pool. The default is 0 — no initial tiny data buffer pool.

Examples

This example shows a TINYDATABUFFERPOOLSIZE statement specifying the number of tiny data buffers to be 500.

```
TinyDataBufferPoolSize   500
```

Usage Notes

- The system will attempt to dynamically allocate 10% more tiny data buffers any time the tiny data buffer free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many tiny data buffers your system is using. To avoid dynamic allocation of tiny data buffers during operation, use the TINYDATABUFFERPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.
- If there are data buffers in the tiny data buffer pool, TCP/IP will use them for applications that exchange small amounts of data, such as the FTP and Telnet servers.
- If you are using Offload, tiny data buffers are used to exchange small amounts of data or control information between the TCPIP virtual machine and the Offload hosts. You should have at least 500 tiny data buffers for TCP/IP to work correctly with the Offload hosts.

Context

- “DATABUFFERPOOLSIZE Statement” on page 63
- “SMALLDATABUFFERPOOLSIZE Statement” on page 124
- *TCP/IP User’s Guide* for the NETSTAT command

TN3270E Statement

Use the TN3270E statement to define client IP addresses and logical unit names that may establish printer sessions and associate a user identifier and virtual address with the session.

```
                 ┌──────────────────────────────────────┐
                 ▼                                      │
►►──TN3270E───────ip_address luname user_id vaddr───────┴──ENDTN3270E──────────────►◄
```

Operands

ip_address
: The IP address of the client who is permitted to establish a TN3270E printer session.

luname
: The logical unit name the client must provide.

user_id
: The user identifier of the virtual machine to be associated with the printer session (for example, RSCS).

vaddr
: The virtual address in the aforementioned virtual machine that is associated with the printer session.

Examples

The following example allows two TN3270E printer sessions to be established:

```
TN3270E
  9.130.58.29  JMARSH  RSCS 360
  9.130.67.224 REFSNID RSCS 361
ENDTN3270E
```

Usage Notes

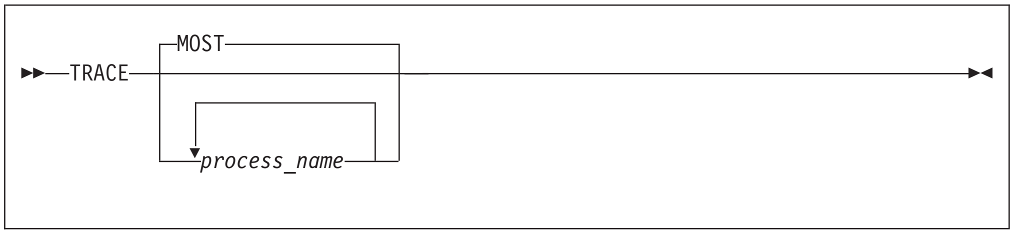
- If the IP address and LU name supplied by a client who is establishing a TN3270E printer session match a specification in the TN3270E statement, the associated user identifier and virtual device address are provided to the Printer Management Exit.
- When a printer session is accepted, a logical printer device is created to represent it.
- A TN3270E printer management exit must be provided to connect the logical printer device to a virtual machine that can create and transmit 3270 data streams to it. See the *TCP/IP Programmer's Reference* for more information about this exit.
- If the OBEYFILE command is used and includes the TN3270E statement, the entire set of printer session definitions is replaced. Existing printer sessions are not affected.

Context

- "INTERNALCLIENTPARMS Statement" on page 105

TRACE Statement

Use the TRACE statement to establish the list of internal TCP/IP stack processes for run-time tracing. The TRACE statement is intended for debugging, in consultation with the IBM TCP/IP for VM support group.



Operands

process_name
The name of a process for which you want run-time trace information. Refer to Table 11 on page 131 for valid process names.

Usage Notes

- The SCREEN configuration statement directs trace messages to the console. The NOSCREEN statement directs them to the file selected by the FILE statement.

  Tracing TCP/IP processes can (and often will) produce a large amount of data. If trace messages are directed to the trace file and the disk containing the trace file becomes full, trace output is terminated.
- To disable run-time tracing, use the NOTRACE configuration statement.

- If a TRACEONLY statement is used, the trace is restricted to the processes that are related to the selected users, devices, or IP addresses.

Context

- "FILE Statement" on page 90
- "LESSTRACE Statement" on page 111
- "MORETRACE Statement" on page 113
- "NOTRACE Statement" on page 114
- "SCREEN Statement" on page 123
- "NOSCREEN Statement" on page 114
- "TRACEONLY Statement"
- *TCP/IP Diagnosis Guide*

Table 11. TCP/IP Process Names. These names are valid for the TRACE, LESSTRACE, MORETRACE, and NOTRACE statements.

ALL	IP-DOWN	RETRANSMIT
ARP	IP-REQUEST	REXMIT
ATM	IP-UP	ROUND-TRIP
CCS	IUCV	SCHEDULER
CETI	IUCVGREETER	SHUT-DOWN
CLAW	IUCVHANDLER	SNMPDPI
CONGESTION	IUCVSIGNON	SOCKET
CONSISTENCY-CHECKER	MONITOR	STATUS-IN
CTC	MOST	STATUS-OUT
DYNROUTING	MULTICAST	TCP
ELANS	NO-PROCESS	TCP-DOWN
EXTERNAL-HANDLER	NONE	TCP-IP
FPSM	NOTIFY	TCP-REQUEST
GIGABIT	OFFLOAD	TCP-UP
HANDLERS	PACKET	TELNET
HCH	PARSE-TCP	TIMER
HCHINCORE	PCCA	TO-1822
ICMP	PING	TOIUCV
IGMP	QDIO	UDP
ILANS	RAWIP	UDPREQUEST
INITIALIZE	RAWIPREQUEST	UDPUP
IO-HANDLER	RAWIPUP	X25ICA
IOCTLROUTE		

Notes:

1. Process names ALL, MOST, NONE, and NO-PROCESS are aliases. The MOST process alias includes all processes except INITIALIZE, SCHEDULER, and TIMER. The alias ALL includes all processes. TRACE NO-PROCESS and TRACE NONE turn off all tracing.
2. Hyphens (-) in process names may be omitted.

TRACEONLY Statement

Use the TRACEONLY statement to restrict TCP/IP stack tracing to particular users, devices, or IP addresses. It is intended for debugging, in consultation with the IBM TCP/IP support group.

```
                    ┌──────────────────┐
                    ▼                  │
►►──TRACEONLY───────┬────────────────┬─┴──ENDTRACEONLY────────────────────►◄
                    ├─user_id────────┤
                    ├─device_name────┤
                    └─internet_addr──┘
```

Operands

user_id
The user identifier of the client you want to trace. In order for a selection to be effective, the client must be either active or included in the PORT, OBEY, INFORM, or PERMIT list.

device_name
The name of the device you want to trace.

internet_addr
The IP address you want to trace, in dotted decimal form.

Examples

- This example shows how to enable selective tracing for user FTPSERVE:

```
FILE REQUEST TRACE A
TRACEONLY FTPSERVE ENDTRACEONLY
TRACE TCPREQUEST
```

- This example terminates selective tracing:

```
SCREEN
NOTRACE
TRACEONLY ENDTRACEONLY
```

Usage Notes

- Specify the processes to be traced using the TRACE, MORETRACE, and LESSTRACE statements. When there is activity related to the selected users, devices, or IP addresses, the trace is enabled. Otherwise, the trace is disabled.
- Certain processes are not relevant for certain entities. For example, the CLAW process is relevant to device tracing; enabling a selective trace for a user will not produce output from the CLAW process, even if the process is among those being traced.
- Use the NOTRACE statement to turn off selective tracing and, if the trace is being written on disk, close the output file.
- To turn off selective tracing and enable tracing for the entire stack, specify the TRACEONLY statement with no operands.
- A maximum of 64 IP addresses can be selected for tracing at one time.

Context

- "FILE Statement" on page 90
- "INFORM Statement" on page 103
- "LESSTRACE Statement" on page 111
- "MORETRACE Statement" on page 113
- "NOSCREEN Statement" on page 114

- “NOTRACE Statement” on page 114

TRANSLATE Statement

Use the TRANSLATE statement to indicate the relationship between an IP address and the network address on a specified link. The TRANSLATE statement is required for HYPERchannel hosts (HCH) on a directly-connected HYPERchannel network. It is also required for X.25 hosts on a non-Defense Data Network (DDN) when using an X.25 ICA. You can also use the TRANSLATE statement, with some limitations, for Ethernet and token-ring hosts that do not support ARP, as well as for ATM networks that do not have an ATMARP server.

```
              ┌──────────────────────────────────────────────────────┐
              ▼                                                      │
►►──TRANSLATE───internet_addr──┬─ATM──────┬──net_addr link_name──┴──────────►◄
                               ├─ETHERNET─┤
                               ├─FDDI─────┤
                               ├─HCH──────┤
                               ├─IBMTR────┤
                               └─X25ICA───┘
```

Operands

internet_addr
The IP address in dotted decimal form for which a translation is specified.

ATM
Indicates that *net_addr* is an ATM address.

ETHERNET
Indicates that *net_addr* is an Ethernet address.

FDDI
Indicates that *net_addr* is an FDDI address.

HCH
Indicates that *net_addr* is a HYPERchannel address.

IBMTR
Indicates that *net_addr* is an IBM Token-Ring address.

X25ICA
Indicates that *net_addr* is an X.25 address.

net_addr
The network address corresponding to *internet_addr* and *link_name*. The format depends on the network type.

- 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 TCP/IP for VM or TCP/IP for MVS system, then *cc* is the read port address (the lower of the two addresses that are attached to the TCPIP virtual machine).

- 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 in DIX Ethernet format, not 802.3 format.
  - For token-ring, the translation table entry should not contain a token-ring source routed bridge path.
- For X25ICA, specify a decimal number up to 15 digits.
- For ATM, specify a 40-digit hexadecimal number.

link_name
A network link name (from a LINK statement). The specified *internet_addr* is translated to the specified *net_addr* only when sending on this link. You can include multiple TRANSLATE statement entries for the same *internet_addr* with different *link_name*s.

Examples

This example shows the TRANSLATE statement for HYPERchannel:

```
Translate
  9.67.51.3    HCH    FF0000006702   HCH1
  9.67.22.4    HCH    FF0000009A05   HCH1
```

Usage Notes

- The first TRANSLATE statement of each configuration file processed replaces the internal translation tables (the ARP table), including information dynamically added by ARP, with the new information. Subsequent TRANSLATE statements in the same file add entries to the table.
- If you are using Offload, do not specify the Offload host links in the TRANSLATE statement. ARP processing is handled by the Offload host. See *TCP/IP Offload Feature for z/VM* book for more information about Offload ARP processing.

  Do not specify the name of an Offload API link as *link_name*.
- Some token ring hardware does not recognize the RFC 1469 mandated functional MAC address (’C00000040000’X) 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 the following special physical addresses:
  - FFFFFFFFFFFF for all rings broadcast.
  - C00000040000 to reset back to the default functional MAC address.

  The following are examples of how to specify each method:
  - All rings:
    ```
    TRANSLATE
      224.0.0.0  IBMTR  FFFFFFFFFFFF linkname
    ```
  - Functional MAC 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 functional MAC address, as it is the default method.

UCBPOOLSIZE Statement

Use the UCBPOOLSIZE statement to set the initial number of UDP control blocks (UCBs). UCBs are used to hold information about open UDP ports. Each one requires 245 bytes.

```
     ┌─UCBPOOLSIZE 100────┐
►►───┤                    ├──────────────────────────────────────────────►◄
     └─UCBPOOLSIZE number─┘
```

Operands

number
The initial number of UCBs in the free pool. The default is 100.

Examples

This example shows a UCBPOOLSIZE statement setting the number of UCBs to the default of 100.

```
UCBpoolSize   100
```

Usage Notes

- The system will attempt to dynamically allocate 10% more UCBs any time the UCB free pool level drops to 5%. You can use the NETSTAT POOLSIZE command to monitor how many UCBs your system is using. To avoid dynamic allocation of UCBs during operation, use the UCBPOOLSIZE statement to initialize the free pool size to the maximum shown by NETSTAT POOLSIZE.

Context

- *TCP/IP User's Guide* for the NETSTAT command

Changing the TCP/IP Configuration with the OBEYFILE Command

The OBEYFILE command allows you to make temporary dynamic changes to the system operation and network configuration without stopping and restarting the TCPIP virtual machine. You can maintain different files that contain a subset of the TCP/IP configuration statements and use OBEYFILE to activate them while TCP/IP is running.

OBEYFILE Command

Use the OBEYFILE command to redefine your TCP/IP configuration without interrupting it. These changes are temporary and may be altered if another OBEYFILE command is issued or when TCP/IP is restarted.

The OBEYFILE command is issued from CMS. It instructs TCP/IP to read a new configuration information file (the *obey file*) while it is running. In order to issue this command, your user ID must be included in the OBEY list in the configuration file. When an unauthorized user issues an OBEYFILE command, audit messages are written to the TCPIP virtual machine console.

Some statements are ignored during OBEYFILE processing and others have limitations. These restrictions are summarized in the Usage Notes section below.

```
                         ┌─TCPIP────┐  ┌─A────────┐
►►──OBEYFILE──filename───┴─filetype─┴──┴─filemode─┴──(─┬──────────┬──────────►
                                                        └─password─┘
►─┬────────────┬──────────────────────────────────────────────────────────►◄
  └─TCP server─┘
```

Operands

filename

filetype

filemode
: The name of a CMS file that contains TCP/IP configuration statements. The file type defaults to **TCPIP** and the file mode defaults to **A**.

password
: The CP minidisk read password. If the password is "ALL" or if the minidisk is protected by an external security manager such as RACF, then omit the password.

TCP *server*
: The user identifier of the TCP/IP virtual machine whose configuration you want to change. If this option is not used, OBEYFILE addresses the TCP/IP machine identified in the `TCPIP DATA` file.

Examples

- In this example, file TRACE TCPIP A contains configuration statements to activate ping tracing and direct trace output to file TCPIP LOGFILE L. The user minidisk containing file TRACE TCPIP is accessed as file mode A and is protected by RACF. The TCPIP virtual machine has been given READ access to this minidisk.

  ```
  FILE TCPIP LOGFILE L
  TRACE  PING
  ```

  The trace is enabled by the command:

  ```
  obeyfile trace
  ```

  Enable the trace for the TCPTEST virtual machine using the command:

  ```
  obeyfile trace (tcp tcptest
  ```

- In this example, NOTRACE TCPIP B contains statements to turn off all tracing, close the current trace file, and route any future trace output to the console. The CP read password for the minidisk accessed as B is "readpw".

  ```
  NOTRACE
  SCREEN
  ```

  The trace is disabled by the command:

  ```
  obeyfile notrace tcpip b (readpw
  ```

Usage Notes

- The obey file must reside on a minidisk. The OBEYFILE command does not support the CMS Shared File System (SFS).
- TCP/IP ignores the CONNECTEXIT, PORT, TN3270EEXIT, and TRANSFORM parameters on any INTERNALCLIENTPARMS statement included in the obey file.
- TCP/IP ignores any MONITORRECORDS statements included in the obey file.
- You cannot add new DEVICE and LINK statements for Offload devices using OBEYFILE, nor can you modify or delete an existing Offload DEVICE or LINK statement.
- When you add or change a configuration statement using OBEYFILE, be aware that the existing statement is replaced. Therefore, the obey file must include the *entire* statement, not just the new or changed portions.

  For example, when you add new LINK statements, all the entries are deleted from the GATEWAY, HOME, BSDROUTINGPARMS, and TRANSLATE statements. Be sure to include the complete entries for the GATEWAY, HOME, BSDROUTINGPARMS, and TRANSLATE statements when adding new LINK statements in the obey file.
- | If you change BSDROUTINGPARMS entries with an OBEYFILE command, issue a subsequent SMSG RECONFIG command to the RouteD server for your changes to take effect. Also, to change the subnet mask for a BSDROUTINGPARMS entry, you must include a (complete) GATEWAY statement in the OBEYFILE file to force TCP/IP to reinitialize its routing tables.
- If you change BSDROUTINGPARMS entries with an OBEYFILE command, issue the SMSG RECONFIG command for RouteD to have the changes take affect. Also, to change the subnet mask for a BSDROUTINGPARMS entry, you must include an empty GATEWAY statement in the OBEYFILE file to force TCP/IP to reinitialize its routing tables
- The following statements cause an error and should not be included in an obey file:

ACBPOOLSIZE	RCBPOOLSIZE
ADDRESSTRANSLATIONPOOLSIZE	SCBPOOLSIZE
CCBPOOLSIZE	SKCBPOOLSIZE
DATABUFFERPOOLSIZE	SMALLDATABUFFERPOOLSIZE
ENVELOPEPOOLSIZE	TCBPOOLSIZE
| FIXEDPAGESTORAGEPOOL	TINYDATBUFFERPOOLSIZE
IPROUTEPOOLSIZE	UCBPOOLSIZE
LARGEENVELOPEPOOLSIZE	

- A successful OBEYFILE command is confirmed by the message:

  ```
  userid has read and obeyed file filename filetype
  ```
- When there is a problem with OBEYFILE, file PROFILE TCPERROR is returned containing a description of the error and an error message is displayed:

  ```
  TCPIP says:  Error in profile file.  Details are in PROFILE TCPERROR
  ```

| Starting and Stopping TCP/IP Services

| Specific TCP/IP services that are managed by server virtual machines other than the TCP/IP stack can also be started and stopped (and in some cases, modified while a server continues operation) on a server-specific basis. For more information, see “Starting and Stopping TCP/IP Servers” on page 30.

Starting and Stopping TCP/IP Services

In order to control the availability of TCP/IP services, there are procedures to start and stop the TCPIP virtual machine (the stack) and its related service virtual machines.

Chapter 5. Defining the TCP/IP System Parameters

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

Configuring the TCPIP DATA File

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

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

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

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

You can specify an optional *system_name* before each configuration statement. The configuration statement is in effect if the *system_name* matches the name of the system determined by the CMS IDENTIFY command. The configuration statement is ignored if the *system_name* does not match the name of the system determined by the CMS IDENTIFY command. The configuration statement is in effect on every system if a *system_name* has not been specified.

Statement Syntax

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

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

ATSIGN Statement

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

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

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

```
            ┌─X’7C’ ─┐
►►──ATSIGN──┴─X’xx’ ─┴──────────────────────────────────────────►◄
```

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

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

```
ATSIGN X'EC'
```

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

Notes:

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

DOMAINLOOKUP Statement

The DOMAINLOOKUP statement specifies what methods are used to resolve host names and in what order these methods are used. By default, name resolution is attempted by asking a question of each name server and then checking for the answer in the HOSTS SITEINFO file. The resolution is complete when the first answer is found. The DOMAINLOOKUP statement can be used to instruct the resolver to use *only* name servers to resolve host names, or to attempt to find an answer in the HOSTS SITEINFO file before any name servers are contacted.

```
          ┌─DOMAINLOOKUP DNS────────┐
►►────────┼─────────────────────────┼──────────────────────────────────────►◄
          ├─DOMAINLOOKUP FILES──────┤
          ├─DOMAINLOOKUP DNSONLY────┤
          └─DOMAINLOOKUP FILESONLY──┘
```

Parameter	Description
DNS	Use both the name servers specified by the NSINTERADDR statements and the HOSTS SITEINFO file to perform host name resolution. Name servers are used before the HOSTS SITEINFO file.
FILES	Use both the HOSTS SITEINFO file and the name servers specified by the NSINTERADDR statements to perform host name resolution. The content of the HOSTS SITEINFO file is used before any name servers.
DNSONLY	Use only the name servers specified by the NSINTERADDR statements to perform host name resolution.
FILESONLY	Use only the HOSTS SITEINFO file to perform host name resolution.

When both the name servers and the HOSTS SITEINFO file are used for searches and the resolver is required to ask more than one question to obtain an answer, each question is asked of each name server, in turn with the HOSTS SITEINFO file being consulted last (assuming the default DNS specification is in use). In other words, the HOSTS SITEINFO file is consulted after the first question has been asked of all name servers. Refer to the “DOMAINSEARCH Statement” on page 142 for more detail about name resolution.

DOMAINORIGIN Statement

The DOMAINORIGIN statement specifies the domain that is appended to the name in the HOSTNAME statement to form the full host name for a system. Case translation is not performed on the domain origin. The DOMAINORIGIN configuration statement must be customized at each site.

The domain provided on the DOMAINORIGIN statement is also used as a DOMAINSEARCH value for the purposes of host name resolution. The rules that govern how host name resolution is performed are described in the “DOMAINSEARCH Statement” on page 142.

```
►►──DOMAINORIGIN──domain──────────────────────────────────────────────────►◄
```

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

DOMAINSEARCH Statement

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

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

```
►►──DOMAINSEARCH──domain──────────────────────────────────────────────►◄
```

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

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

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

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

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

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

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

If one issues this PING command:

```
ping tootsie
```

these questions are sent to the name server:

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

For this command:

```
ping tootsie.comedy
```

these questions are sent to the name server:

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

If this command is issued:

```
ping tootsie.comedy.
```

the questions sent to the name server are:

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

Finally, if this PING is performed:

```
ping junk
```

these questions are sent to the name server:

```
junk.tragedy.movie.edu
junk.comedy.movie.edu
junk.suspense.movie.edu
junk.movie.edu
junk
(the search fails)
```

HOSTNAME Statement

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

```
►►──HOSTNAME──hostname──────────────────────────────────────────────►◄
```

Parameter	Description
hostname	Specifies the TCP host name of the VM host.

NSINTERADDR Statement

The NSINTERADDR statement defines the internet address of a name server in dotted-decimal format. This parameter can be repeated, without limit, to specify the internet addresses of alternative name servers. Connections to the name servers are attempted in the order they appear in this configuration file. If NSINTERADDR statements are not coded in the TCPIP DATA file, the resolver looks up all domain names in the site hosts table see, “Chapter 6. Configuring the HOSTS LOCAL File” on page 151, the resolver does not attempt to use a name server.

```
►►──NSINTERADDR──internet_address────────────────────────────────────►◄
```

Parameter	Description
internet_address	Specifies the internet address of a name server.

NSPORTADDR Statement

The NSPORTADDR statement specifies the port of the name server.

```
     ┌─NSPORTADDR 53──────────┐
►►───┼────────────────────────┴────────────────────────────────────────►◄
     └─NSPORTADDR nsportaddr──┘
```

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

RESOLVERTIMEOUT Statement

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

```
     ┌─RESOLVERTIMEOUT 30─────────────┐
►►───┼────────────────────────────────┴────────────────────────────────►◄
     └─RESOLVERTIMEOUT timeoutvalue───┘
```

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

RESOLVERUDPRETRIES Statement

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

```
       ┌─RESOLVERUDPRETRIES 1─────┐
►►─────┤                          ├──────────────────────────────────────────►◄
       └─RESOLVERUDPRETRIES limit─┘
```

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

RESOLVEVIA Statement

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

```
       ┌─RESOLVEVIA UDP─┐
►►─────┤                ├──────────────────────────────────────────────────►◄
       └─RESOLVEVIA TCP─┘
```

Parameter	Description
TCP	Specifies that the protocol is TCP.
UDP	Specifies that the protocol is UDP. The default protocol is UDP.

SMTPSERVERID Statement

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

```
►►──SMTPSERVERID──user_id──┬──────────────┬─────────────────────────────────►◄
                           └─AT──node_id──┘
```

Parameter	Description
user_id	Specifies the user ID of the SMTP server virtual machine for either the local system, or for a network gateway system.
AT *node_id*	Specifies the RSCS node ID of the location of the SMTP server gateway. The local node is used when a node ID is not specified.

Usage Notes

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

TCPIPUSERID Statement

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

```
   ┌─TCPIPUSERID TCPIP───┐
►►─┼─────────────────────┼──────────────────────────────────────────►◄
   └─TCPIPUSERID user_id─┘
```

Parameter	Description
user_id	Specifies the user ID of the TCPIP virtual machine. TCPIP is the default user ID.

TRACE RESOLVER Statement

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

```
►►──TRACE──RESOLVER─────────────────────────────────────────────────►◄
```

UFTSERVERID Statement

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

```
►►──UFTSERVERID──user_id────────────────────────────────────────────►◄
```

Parameter	Description
user_id	Specifies the user ID of the UFT server virtual machine for the local system. A user ID of * has special meaning and indicates the RSCS virtual machine ID that is returned in the IDENTIFY command should be used.

Usage Notes

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

USERDATA Statement

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

```
►►──USERDATA──┬───────────────────┬──ENDUSERDATA──────────────────────►◄
              │ ┌───────────────┐ │
              │ ▼               │ │
              └───parameter─────┴─┘
```

Parameter	Description
parameter	Specifies an arbitrary, user-defined value.

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

VMFILETYPE Statement

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

```
                                     (1)                (1)
                          ┌─,TRANSLATE=NO──────┐ ┌─,LINES=NONE──────────┐
►►──VMFILETYPE──extension─┼────────────────────┼─┼──────────────────────┼──────►◄
                          └─,TRANSLATE=─┬─YES─┬┘ │                 (2)  │
                                        └─NO──┘  └─,LINES=──┬─NL───┬────┘
                                                            ├─NONE─┤
                                                            └─CMS──┘
```

Notes:

1 Default when no VMFILETYPEDEFAULT statement exists.

2 Used by only the NFS server.

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

Parameter	Description
extension	Defines a file type extension for which translation and line values are being defined; *extension* is a required parameter. The specified file type can begin or end with an asterisk (*) to “wildcard” that file type.
TRANSLATE=YES	
TRANSLATE=NO	Indicates whether EBCDIC-ASCII translation is performed when files with the specified *extension* are processed. Specify **TRANSLATE=YES** if EBCDIC-ASCII translation is to be performed, or **TRANSLATE=NO** if translation should not be performed. The **TRANSLATE=** parameter is optional. If this parameter is omitted, the default established by the VMFILETYPEDEFAULT statement is used; if no such statement exists, **TRANSLATE=NO** is assumed.

LINES=NL

LINES=NONE

LINES=CMS

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

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

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

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

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

Usage Notes

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

VMFILETYPEDEFAULT Statement

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

```
   ┌─VMFILETYPEDEFAULT TRANSLATE=NO,LINES=NONE──────────────────────────────┐
►►─┼────────────────────────────────────────────────────────────────────────┼─────►◄
   │                   ┌─TRANSLATE=NO──────┐       (1)   ┌─LINES=NONE─────────┐ │
   └─VMFILETYPEDEFAULT─┼───────────────────┼─,──────────┼────────────────────┼─┘
                       └─TRANSLATE=──┬─YES─┘             │              (2)   │
                                     └─NO──┘             └─LINES=──┬─NL───┬───┘
                                                                   ├─NONE─┤
                                                                   └─CMS──┘
```

Notes:

1 Required only when following another parameter.

2 Used by only the NFS server.

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

Parameter **Description**

TRANSLATE=YES

TRANSLATE=NO

Indicates whether EBCDIC-ASCII translation is performed when a file is processed. Specify

TRANSLATE=YES if EBCDIC-ASCII translation is to be performed, or **TRANSLATE=NO** if translation should not be performed.

The **TRANSLATE=** parameter is optional. If this parameter is omitted, **TRANSLATE=NO** is assumed.

LINES=NL

LINES=NONE

LINES=CMS

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

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

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

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

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

Testing the TCP/IP System Configuration

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

```
►►──HOMETEST──────────────────────────────────────────────────►◄
```

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

Note: Verify that the TCPIP virtual machine has been started before you use the HOMETEST statement.

Chapter 6. Configuring the HOSTS LOCAL File

A HOSTS LOCAL file contains descriptions of local host entries in the HOSTS format. Any domain name or IP address specified in this file is accessible for use on your network. The HOSTS LOCAL file is used to create the site table. The site table enables name resolution and reverse name resolution without using a domain name server.

A sample file, HOSTS SLOCAL, is supplied with the TCP/IP distribution tapes on the TCPMAINT 592 disk. You can copy the file to the TCPMAINT 198 disk and customize it for use at your site. Each site is unique and requires customized statements.

For example, the NETSTAT and TRACERTE commands use the site tables for inverse name resolution rather than the domain name server. You must still create the HOSTS ADDRINFO and HOSTS SITEINFO files for NETSTAT to execute correctly. These files only need to be created once. They do not have to contain all the hosts in the internet, nor do they have to be updated regularly.

Statement Syntax

The HOSTS LOCAL file can contain three types of entries:
- Host Statement
- Network Statement
- Gateway Statement

One line of the HOSTS LOCAL file is used for each distinct host and ends with four colons. Each line has three essential fields, separated by colons:
- The keyword HOST, NET, or GATEWAY
- A comma-separated list of internet addresses for that host
- A comma-separated list of fully qualified names for that host.

Host and gateway entries also have three optional fields.

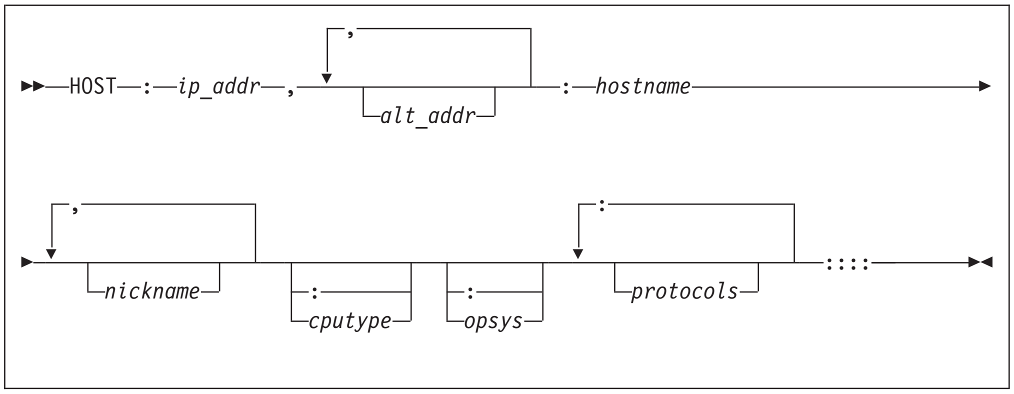
For example, if you have two local hosts, LOCAL1 (internet addresses 192.6.77.4 and 192.8.4.1) and LOCAL2 (with an alias LOCALB and internet address 192.6.77.2), append the following lines to the HOSTS LOCAL file:

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

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

HOST Statement

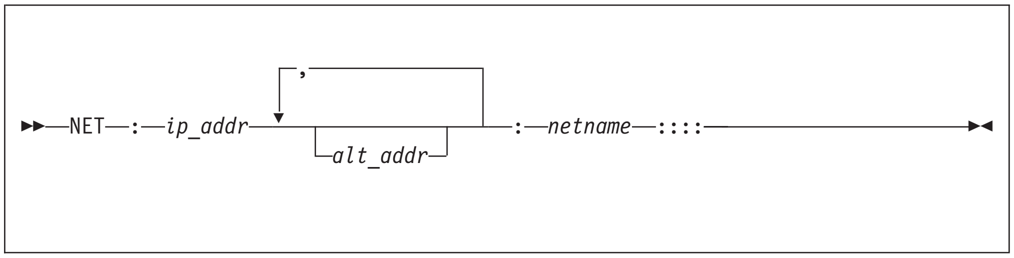
The HOST statement specifies the address and network name of a host that is accessible to TCP/IP.



Parameter	Description
ip_addr	Specifies the internet address of the host.
alt_addr	Specifies the alternative internet address of the host.
hostname	Specifies the fully qualified name of the host. The name can be a maximum of 24 characters and can include the minus sign (–) and a period (.). Periods can only be used to delimit components of a domain name. No blank or space characters are allowed. The *hostname* is not case sensitive. The first character must be an alpha character and the last character cannot be a minus or period.
nickname	Specifies the fully qualified nickname of the host.
cputype	Specifies the machine type or processor type.
opsys	Specifies the operating system of the gateway.
protocols	Specifies the protocols that are supported by the gateway.

NET Statement

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



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

Building the Hosts Local Site Table

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

The following steps describe how to generate the HOSTS SITEINFO and HOSTS ADDRINFO files.

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

   Note: The HOSTS LOCAL table may not contain more than 199,200 site names or 69,500 IP addresses. Sites that require more entries should consider using Domain Name Server (DNS) services.
3. Run the MAKESITE program to produce the HOSTS SITEINFO and HOSTS ADDRINFO files on file mode A disk from the newly modified HOSTS LOCAL file. The virtual machine size required to process a HOSTS LOCAL file with the maximum entries must be large enough to allocate approximately 15M of contiguous storage.

   ```
   ►►──MAKESITE──────────────────────────────────────────────────►◄
   ```

   Note: If you are processing the maximum number of entries, it is recommended that you have defined a *large* virtual machine, 64M. This can take a significant amount of time (up to two hours) to complete this task.
4. If the HOSTS SITEINFO and HOSTS ADDRINFO files exist on the TCPMAINT 592 disk, rename them, for example, to HOSTS SITEOLD and HOSTS ADDROLD. These files are currently accessed by other TCPIP users on the system, and should not be deleted immediately.
5. Copy the new HOSTS ADDRINFO and HOSTS SITEINFO files from your A disk to the TCPMAINT 592 disk.

   Note: The NET statement is not used by IBM-provided TCP/IP applications. It is provided for those applications that use the `getnetent()` socket call.
6. Run the TESTSITE program to test the correctness of the HOSTS SITEINFO file. This step is optional.

   ```
   ►►──TESTSITE──────────────────────────────────────────────────►◄
   ```

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

Chapter 7. Configuring the ROUTED Server

This chapter describes how to configure the RouteD server. It explains RouteD's use of the Routing Information Protocol (RIP) to help you decide if this server is suitable for your network. It also describes configuring RouteD Virtual IP Addressing (VIPA), which gives you the ability to route around interface, device, and network failures. Examples for various configurations are given.

Note: This chapter uses the term "gateway" rather than the more precise term "router" in accordance with the established Routing Information Protocol standards.

Understanding RouteD

The routing daemon is a server that implements the Routing Information Protocol (RIP). It provides an alternative to static TCP/IP gateway definitions. When configured properly, the z/VM host running with RouteD becomes an active RIP router in a TCP/IP network. The RouteD server dynamically creates and maintains the network routing tables using RIP, which eliminates the need to manually configure and maintain a routing table. RIP defines how gateways and routers periodically broadcast their routing tables to adjacent nodes. This is what enables the RouteD server to update the host routing table. For example, the RouteD server can determine if a new route has been created, if a route is temporarily unavailable, or if a more efficient route exists for a given destination.

Routing Information Protocol

The Routing Information Protocol (RIP), as specified in RFC 1058 (RIP version 1) and RFC 1723 (RIP version 2), 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 RouteD server, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network.

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 zero hops away from directly connected networks, one hop away 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 RouteD server broadcasts routing information to the 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 is not received from a gateway for 180 seconds (3 minutes), RouteD assumes the gateway is down and sets all the routes through that gateway to a metric of 16 (infinity). If an update is not received from such a gateway in after an additional 120 seconds (2 minutes), RouteD deletes all routes known through that gateway.

During the intervals specified by the *interface.scan.interval* and *interface.poll.interval* values on the OPTIONS statement, RouteD 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, RouteD receives its own broadcasted packets over the interfaces, provided that the interfaces are active. Other networks, such as point-to-point links, cannot be managed by RouteD unless a RouteD server is running on the host on the other end of the link. If the other host is not running RouteD, the RouteD server does not receive updates over the link and deletes all the routes over the point-to-point link, as described earlier. For more information, see "Configuring a Passive Route" on page 171.

RouteD requires routers on interfaces that do not support broadcasting (for example, the CTC interface) to be active gateways, since it uses link-level broadcasting to send routing updates. For more information on how to manage CTC networks, see "Active RIP Routes" on page 158. The RouteD server never sends routing updates to the CTC network, because it uses link-level broadcasting to send routing updates.

RIP Version 2

RIP Version 2 is an extension of RIP Version 1 and provides the following features:

Route Tag
The Route Tag field is used to hold an attribute that is assigned to a route and must be preserved and readvertised with the route. The route tag field may be used to propagate information about a route that was acquired from an EGP (Exterior Gateway Protocol). RouteD does not generate route tags, but will preserve them in received routes and readvertise them when necessary.

Variable Subnetting
Variable-length subnet masks are included in routing information so that dynamically-added routes to destinations outside subnetworks or networks are reachable.

Next Hop
The purpose of the Next Hop field is to eliminate routing packets through extra hops in the network. An address in the next hop field indicates an "immediate" next hop, as opposed to the current multi-hop route. RouteD does not generate immediate next hops, but will preserve them if they are included in RIP packets.

Multicast for RIP-2 Packets
An IP multicast address 224.0.0.9 is reserved for RIP-2 packets. RouteD | supports sending and receiving multicasted RIP Version 2 packets on this address.

Authentication of RIP-2 Packets for Routing Update Security
Authentication uses passwords that can be configured to be included in outgoing RIP-2 packets for authentication by adjacent routers as routing update security. Likewise, incoming RIP-2 packets are checked against a local authentication password. Any outgoing or incoming RIP-1 packets are not authenticated. For maximum security, configure RouteD so that it will supply and receive RIP-2 packets only and supply an authentication password. The authentication passwords are configurable on per-interface (router) basis.

Routing Control

Routing controls are provided to selectively control which versions of RIP packets are to be sent or received over network interfaces. The switches should be set based on the routing capabilities of the network and are configurable on a per-interface (router) basis.

Supernetting Support

Supernetting is part of Classless InterDomain Routing (CIDR) function. It 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 RIP-2 messages. If local supernet routes are defined for RouteD, they will be advertised to adjacent routers. Local supernet routes are generated by RouteD for interfaces with subnet masks that are less than the value of the network class mask in value.

ATM Network Considerations

RouteD 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 RouteD. However, RouteD cannot be used to manage ATM network interfaces operating in native mode. In this case, static routes to the ATM network must be configured via the GATEWAY statement in PROFILE TCPIP. See “GATEWAY Statement” on page 92 for more information on how to add static routes for ATM networks. Also, to prevent RouteD from adding or deleting routes to the ATM network through other interfaces, it is necessary to add blocking statements in the Routed gateways file for all other interfaces on the VM system. All IP subnet addresses in the ATM network should be blocked for all gateways on the VM system. For more information on blocking routes, see “RIP Input/Output Filters” on page 159.

RIP Routes

TCP/IP routes that are added and maintained by RouteD are known as RIP routes. Only RouteD (or a similar application) can add or delete RIP routes. Furthermore, if RouteD is terminated, RIP routes are not deleted. To help maintain the integrity of the routing table, at initialization, RouteD deletes all RIP routes from the stack routing table and then adds RIP routes based on BSDROUTINGPARMS settings and the optional ETC GATEWAYS file.

RouteD RIP Route Types

RouteD supports four types of RIP Routes:
- Passive
- Permanent
- External
- Active

These route types are used to instruct RouteD on how to support gateways defined in the ETC GATEWAYS file.

Passive RIP Routes

Passive RIP routes are known by both TCP/IP and RouteD. Information about passive routes is put in TCP/IP’s and RouteD’s routing tables. A passive entry in RouteD’s routing table is used as a place holder 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, therefore 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 target 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.

Permanent Routes

Permanent routes are like passive routes except that they are propagated to other networks. This may be necessary in certain configurations, such as when a destination cannot propagate its own routing information. However, because it is possible to create routing loops and because recoverability is poor, permanent routes should be used with care. Passive routes should be used instead of permanent routes, whenever possible.

External RIP Routes

External RIP routes are known by RouteD but not by TCP/IP. External routes are managed by other protocols such as the External Gateway Protocol (EGP). The RouteD server needs to know not to interfere with these routes and not to delete them.

An external entry exists in RouteD’s routing tables as a place holder to prevent a route from being overwritten by a competing RIP route. External routes are not propagated. RouteD does not manage external routes. Therefore, RouteD only knows that there is an existing route to a host or network and one that is not known to TCP/IP.

External routes should be used when the local host is running with some type of non-RIP routing protocol that 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 RIP Routes

An active RIP route is treated as a network interface.

Active gateways are routers that are running RIP but are reached by a medium that does not allow broadcasting and is not point-to-point. RouteD normally requires that routers be reachable via broadcast addresses or via a point-to-point link. If the interface is neither, then an active gateway entry can add the gateway to RouteD’s interface list. RouteD will treat the active gateway as a network interface. Note that the active gateway must be directly connected.

Active routes should be used when the foreign router is reachable over a non-broadcast, non-point-to-point network and is directly connected to the local host.

RouteD will communicate with active routers by point-to-point transmissions to the gateway address. Routes are not added to neither RouteD nor the TCP/IP routing table immediately. They are added and propagated normally when route advertisements arrive from an active gateway. The only effect of an active gateway statement is to bypass the requirement for broadcast communication on true 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 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.

RouteD RIP Route Summary

Table 12. RouteD Gateway Summary

	Propagated?	Known by TCP/IP?	Known by RouteD?	Timeout?
Passive	No [2]	Yes	Yes	No
Permanent	Yes	Yes	Yes	No
External	No	No	Yes	No
Active	Yes	Yes	Yes	Yes

RIP Input/Output Filters

The RIP input/output filters provide routing table manipulation and routing control. The filters are provided by RouteD 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 Broadcasting Switch
6. Interface RIP Switch
7. Default Route Only Broadcasting Switch
8. Virtual Route Only Broadcasting Switch
9. Default and Virtual Routes Only Broadcasting Switch
10. Local (directly-connected) Routes Only Broadcasting Switch
11. Triggered Updates Only Broadcasting Switch
12. Gateway NoReceiving

For more information on these RIP input/output filters, see the RouteD procedure parameters in "ROUTED Command" on page 169 and the options statement in "Step 5: Configure the ETC GATEWAYS File (Optional)" on page 162.

Configuration Process

The steps to configure RouteD are as follows:

1. Create the RouteD configuration file.
2. Update PORT, BSDROUTINGPARMS, and GATEWAY statements in the TCPIP profile.
3. Update the ETC SERVICES file.
4. Update the DTCPARMS file (Optional).
5. Configure the ETC GATEWAYS file (Optional).

2. Except directly-connected passive routes. Directly-connected passive routes are propagated to other network interfaces for network connectivity. A directly-connected passive route is one whose gateway address is one of the local interfaces in the HOME list or is one of the offload interfaces.

Note: If a default route is to be defined to a destination gateway or router, configure a default route in this ETC GATEWAYS file.

Step 1: Create the RouteD Configuration File (ROUTED CONFIG)

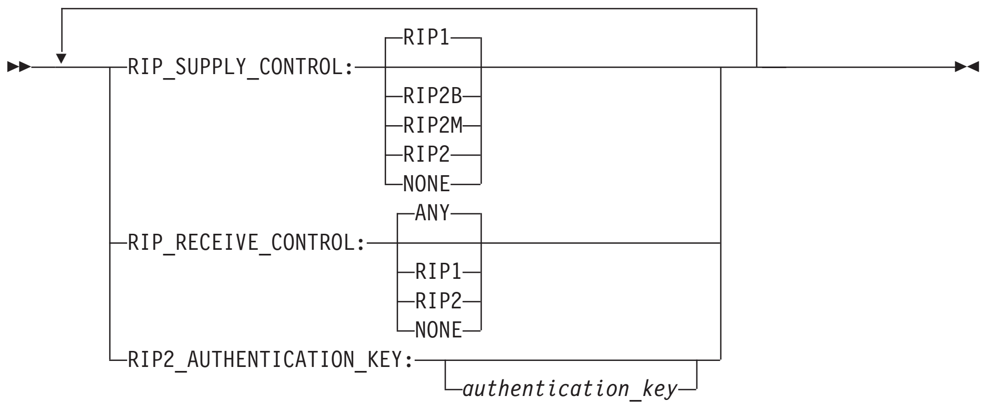
RouteD supports sending and receiving both RIP version 1 and RIP version 2 packets. A new configuration file determines the mode of operation. A sample of this configuration file is shipped as ROUTED SCONFIG on TCPMAINT 591. It should be copied to TCPMAINT 198, customized, and renamed to ROUTED CONFIG.

Configuration Note: In the absence of a ROUTED CONFIG file, the server will use the default settings;
- RIP_SUPPLY_CONTROL: **RIP1**
- RIP_RECEIVE_CONTROL: **ANY**
- RIP2_AUTHENTICATION_KEY: (no key is defined)

Syntax Rules

The following is a list of syntax rules for the RouteD config file:
- Keywords (for example, RIP_SUPPLY_CONTROL) are treated as if they were entered in all capital letters.
- Blank records may be used to improve readability.
- Configuration statements must start and end on the same line.
- All comments must be preceded by a ; (semicolon). A comment may follow a complete keyword and data specification on a record, or it may occupy a complete record.

Following are the options that can be included in the ROUTED CONFIG file:



RIP_SUPPLY_CONTROL:
A constant. Specifies that the keyword following will be used as the RIP supply control for all interfaces. Possible supply controls are as follows:

RIP1	Unicast/Broadcast RIP Version 1 packets (Default)
RIP2B	Unicast/Broadcast RIP Version 2 packets.

Note: Care must be taken when using the RIP2B option since host route misinterpretations by adjacent routers running RIP Version 1 can occur.

RIP2M	Multicast RIP Version 2/Broadcast RIP Version 1 packets (Migration)
RIP2	Multicast RIP Version 2 packets
NONE	Disable sending RIP packets

RIP_RECEIVE_CONTROL:
A constant. Specifies that the keyword following will 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

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 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 broadcasts for authentication by adjacent routers running RIP Version 2. A null key (no key 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.

Step 2: Specify Configuration Statements in PROFILE TCPIP

You should include ROUTED in the AUTOLOG statement of the PROFILE TCPIP configuration file as a startup procedure. The ROUTED server is then automatically started when you initiate TCPIP. Verify that the following statements have been added to the PROFILE TCPIP file:

```
AUTOLOG
  ROUTED  0
```

In the PORT statement, you must reserve port 520 UDP for ROUTED. Verify that the following statements have been added to the PROFILE TCPIP:

```
PORT
  520 UDP ROUTED
```

Use the OBEY statement to include the ROUTED user ID in the OBEY list. This allows the RouteD server to change routing information. Verify that the following statements have been added to the PROFILE TCPIP. Your OBEY statement can list other privileged users, depending on your configuration.

```
OBEY
  ROUTED
ENDOBEY
```

The GATEWAY and BSDROUTINGPARMS statements are used to configure routing information in the PROFILE TCPIP configuration file. The GATEWAY statement is used to configure static routes. The BSDROUTINGPARMS statement is used to define the characteristics of each virtual and physical link. Routes can be added with the GATEWAY statement, but are not added to RouteD's internal routing tables. These routes are not broadcast to other RIP servers. The GATEWAY statement describes static routes to be added to the routing table. If RouteD is running, then routes can still be defined using the GATEWAY statement, but these routes are not used by RouteD and are not advertised using RIP. If you need to add routes to gateways that do not support RouteD, use the ETC GATEWAYS file. If you are not using RouteD, then do not use the BSDROUTINGPARMS statement to configure routes.

Code the following on the ASSORTEDPARMS statement in PROFILE TCPIP:

```
ASSORTEDPARMS
  IGNOREREDIRECT
  SOURCEVIPA
ENDASSORTEDPARMS
```

Do not specify the no forwarding (NOFWD) option. To enable variable subnetting, add the VARSUBNETTING option. If RouteD will be used to either send or receive RIP version 2 packets, the VARSUBNETTING option must be specified in the TCPIP profile. See "ASSORTEDPARMS Statement" on page 50 for descriptions and examples.

Note: The specification of the VARSUBNETTING option requires that the ASSORTEDPARMS statement be placed before the Gateway and BSDRoutingParms statements.

Step 3: Update the ETC SERVICES file

The ETC SERVICES file contains the relationship between services and port numbers. The file must exist for ROUTED to run. Ensure that the following line is added to the ETC SERVICES file:

```
router       520/udp     route routed
```

Note: In the above example, the default well-known port (520) is used.

You should compare the port number configured in the ETC SERVICES file to the port number configured for the ROUTED server in the PROFILE TCPIP file to ensure that the port numbers are the same.

Step 4: Update the DTCPARMS File (Optional)

The TCP/IP server initialization program searches for the configuration definitions for each server. The tags that will directly affect the ROUTED virtual machine are:

```
:Nick.ROUTED
  :Parms.
```

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used. For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the RouteD server if you need to specify any of the RouteD server input parameters. See "ROUTED Command" on page 169 for a complete list.

Step 5: Configure the ETC GATEWAYS File (Optional)

The RouteD server queries the network and dynamically builds routing tables from routing information transmitted by other routers that are directly connected to the network. The ETC GATEWAYS file is used to identify networks or hosts that are not on a directly connected network, but are still accessible through other gateways. The ETC GATEWAYS file is stored on TCPMAINT 198, which the RouteD server reads when it starts.

Note: The ETC GATEWAYS file is not related to the GATEWAY statement used in PROFILE TCPIP.

Passive Routes

Passive routes are not propagated; they are known only by the local host. Routes to the gateway are maintained in the local routing tables indefinitely and are local only to this ROUTED server.

You should use passive routes when the host or network is not running RIP, but you want to add a route to RIP. Passive routes should also be used when adding a default route, because this ensures that the route does not time out.

Permanent Routes

Permanent routes are like passive routes except they are propagated to other networks. This may be necessary in certain configurations, such as when a destination cannot propagate its routing information. However, because it is possible to create routing loops and because recoverability is poor, permanent routes should be used with care. Passive routes should be used instead.

External Routes

External routes are routes that should never be added to the routing table. They are not propagated. The ROUTED server discards any routes for this destination that it receives from other RouteD servers. An external entry exists in RouteD's tables to prevent the route from being overwritten by a competing RIP route.

You may need the external gateway statement for security purposes to prevent your RouteD server from adding certain routes. External routes should be used when the local machine is running some type of non-RIP routing protocol that dynamically changes TCPIP's routing tables.

Active Routes

Active gateways are routers that are running RIP, but are reached through a medium that does not allow broadcasting and is not point-to-point, such as HyperChannel. RouteD normally requires that routers be reachable through broadcast addresses, or through a point-to-point link. If the interface is neither, then an active route entry can add the gateway to RouteD's interface list. Note that the active gateway must be directly connected.

Active routes 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 machine.

RouteD will communicate with active routes by point-to-point transmissions to the gateway address. Routes are not added to neither RouteD's nor TCPIP's tables immediately, but when route advertisements arrive from the active gateway, they will be added and propagate as normal. The sole effect of an active gateway statement is to bypass the requirement for broadcast communication or true point-to-point links. Note that non-broadcast, non-point-to-point interfaces that are NOT active gateways are assumed to be loopback interfaces to the local machine. Also note that while a route to an active gateway may timeout, the interface entry is never removed. If transmissions resume, the new routes will still be to active gateways.

Syntax Rules

- Keywords can be specified in mixed case.
- Blanks and comments are supported in the ETC GATEWAYS file. Comments are identified by a semicolon in column 1.
- The ETC GATEWAYS definitions are comprised of a routing statement or OPTION statement, or both. The following describes the syntax for the routing statement. The description of the syntax for the OPTION statement is described on page 166.

The syntax for the routing statement for the ETC GATEWAYS file follows:

```
       ┌──────────────────────────────────────────────────────────────────────┐
       ▼                                                                      │
►►─────┬─net────┬─┤ Routing Details ├──┬───────────┬──┬─────────────────────┬─┴──────────►◄
       ├─host───┤                      ├─passive───┤  └─mask ─┬────────────┬┘
       └─active─┘                      ├─permanent─┤          └─subnetmask─┘
                                       ├─external──┤
                                       └─active────┘
```

Routing Details:

```
├──name1──gateway──┬─name2─┬──metric──hop count──────────────────────────────────────┤
                   └─=─────┘
```

net
Indicates the route goes to a network.

host
Indicates the route goes to a specific host.

active
Indicates 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 broadcasting 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. *name1* must be specified as "active" if this is for an active gateway. The last entry in the file must specify an active gateway.

gateway
A constant. The parameters that follow this keyword identify the gateway or router for this destination.

name2
Indicates either a symbolic name or the IP address of the gateway router for this destination, or an equals sign (=) to indicate that the gateway is the same as the destination network specified by *name1*. If an IP address is specified, it must be in the standard dotted decimal notation. All numbers will be interpreted as decimal values only.

metric
A constant. The value that follows this keyword is the hop count to the destination host or network.

hop count
: The hop count to this destination. This number is an integer from 0 to 15.

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 RouteD server. Passive gateway entries are not included in any routing information that is transmitted. Directly connected passive routes are included in routing information that is transmitted.

permanent
: A permanent gateway exchanges passive type routing information through RouteD. This may be necessary in certain configurations where a destination cannot propagate its own routing information.

external
: An external gateway parameter indicates that entries for this destination should never be added to the routing table. The RouteD 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 broadcasting 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, RouteD will default the subnetwork mask to an interface subnetwork mask that matches the route's network. If there is no interface match, the network class mask for the route is used.

Note: For more information on passive, external, and active gateways, see "RouteD RIP Route Types" on page 157.

The following example shows the contents of an ETC GATEWAYS file 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`, which 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`, which 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 routes received from other RouteD 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 ETC GATEWAYS file follows:

```
     ┌──────────────────────────────────────────────────────────────────────┐
     ▼                                                                      │
►►───OPTIONS──┬─gateway ip_addr ─┬──────────────┬──────────────────────────┬─┴───────────►◄
              │                  ├─block────────┤                          │
              │                  ├─noreceive────┤                          │
              │                  └─none─────────┘                          │
              ├─interface.poll.interval timer_value ───────────────────────┤
              ├─interface.scan.interval timer_value ───────────────────────┤
              ├─interface name ip_addr block destination ──────────────────┤
              ├─interface name ip_addr forward destination fmask fmask ────┤
              ├─interface name ip_addr forward.cond destination fmask fmask ┤
              ├─interface name ip_addr noforward destination fmask fmask ──┤
              ├─interface name ip_addr none────────────────────────────────┤
              ├─interface name ip_addr noreceive destination fmask fmask ──┤
              ├─interface name ip_addr passive ────────────────────────────┤
              ├─interface name ip_addr ripon ──────────────────────────────┤
              ├─interface name ip_addr ripoff ─────────────────────────────┤
              ├─interface name ip_addr receive destination fmask fmask ────┤
              ├─interface name ip_addr receive.cond destination fmask fmask ┤
              ├─interface name ip_addr supply off──────────────────────────┤
              ├─interface name ip_addr supply on───────────────────────────┤
              ├─interface name ip_addr key ─┬───────────────────────┬──────┤
              │                             └─authentication_key ───┘      │
              ├─interface name ip_addr nokey───────────────────────────────┤
              ├─interface name ip_addr supply.control─┬─────────┬──────────┤
              │                                       ├─RIP1────┤          │
              │                                       ├─RIP2B───┤          │
              │                                       ├─RIP2M───┤          │
              │                                       ├─RIP2────┤          │
              │                                       └─NONE────┘          │
              └─interface name ip_addr receive.control─┬─────────┬─────────┘
                                                       ├─RIP1────┤
                                                       ├─RIP2────┤
                                                       ├─ANY─────┤
                                                       └─NONE────┘
```

gateway
A constant. The value that follows this keyword identifies the gateway or router.

ip_addr
Specifies the internet address of the interface associated with the interface name. When the **none** suboption is specified for the **gateway** option, *ip_addr* can be specified as an asterisk (*) to signify all internet addresses or all gateway addresses.

block For the gateway option, specifies that routing table broadcasts from this gateway are to be ignored. This option is provided as a RIP input filter.

noreceive
See description for block.

none For the **gateway** option, specifies that any RIP filter options for this gateway are to be turned off or reset. If *ip_addr* is specified as an asterisk (*), all gateway entries with gateway options will be cleared.

interface.poll.interval
Specifies the time interval in seconds for the interface poll interval. RouteD

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.scan.interval
Specifies the time interval in seconds for the interface scan interval. RouteD 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 with the OBEYFILE command.

timer_value
The range is from 30 to 180 seconds in multiples of 30 seconds. The default is 60 seconds.

interface
A constant

| *name* Specifies the name of the interface, as used in the HOME list. When
| the **none** suboption is specified for the **interface** option, *name* can be
| specified as an asterisk (*) to signify all internet addresses or all
| gateway addresses.

ip_addr
Specifies the internet address of the interface associated with the
| interface name. When the **none** suboption is specified for the interface
| option, *ip_addr* can be specified as an asterisk (*) to signify all internet
| addresses or all gateway addresses.

block For the interface option, specifies that the *destination* route in the received broadcasts for this interface is 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 that can be used to filter out all routes broadcast over an interface and allow routes with specified forward and receive filter to be used.

fmask Specifies an optional filter mask that can be used to apply the filter to multiple routes in just a single options statement. The *fmask* is NOT a subnet mask; it is merely used to filter out multiple routes in a single options statement.

forward
Specifies that the *destination* route in the routing table broadcasts 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 in the routing table broadcasts is to be forwarded to this interface only when the interface is active. In case of an interface outage, RouteD will include the *destination* route in the

routing table broadcasts to other active interfaces. 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 routing table is not to be forwarded. This option is provided as a RIP output filter.

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, RouteD will include the *destination* route in the routing table broadcasts to other active interfaces. This option is provided as a RIP input filter.

ripoff Specifies that RIP is disabled for this interface. RouteD will not broadcast and will ignore routing 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 broadcasting is disabled for this interface. RouteD will not broadcast, but continues to receive routing updates. This option is provided as a RIP output filter.

supply on
Specifies that broadcasting 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 *ip_addr* is specified as an asterisk (*), all gateway entries with gateway options will be cleared.

authentication_key
An authentication key containing up to 16 characters to be used for this interface, which is used to override the RouteD config file setting. The key will start with the first character and end at the last character in the string on the line. A null key (no key is specified) indicates that authentication is disabled for the interface.

nokey Specifies that authentication is disabled for this interface, even though the router-wide specification from the RouteD config file is defined.

supply.control
A constant. Specifies that the keyword following is to be used as the RIP supply control for this interface and is used to override the RouteD profile setting. Possible supply controls are as follows:

RIP1	Unicast/Broadcast RIP Version 1 packets (Default)
RIP2B	Unicast/Broadcast RIP Version 2 packets.

Note: For more information about using RIP2B, see Usage Note Number 5

RIP2M	Unicast RIP Version 2/Broadcast RIP Version 1 packets (Migration)
RIP2	Unicast RIP Version 2 packets
NONE	Disable sending RIP packets

receive.control
A constant. Specifies that the variable following is to be used as the RIP receive control for this interface and is used to override the RouteD 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 an ETC GATEWAYS file:

```
options gateway 9.2.1.4 noreceive
options gateway 9.2.1.4 none
options gateway * none
options interface.poll.interval 15
options interface.scan.interval 90
options interface ETH1 10.1.1.1 passive
options interface ETH1 10.1.1.1 supply off
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 interface TR1 9.67.112.25 block 9.1.0.0
options interface TR1 9.67.112.25 supply.control rip1
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 9.1.1.1 forward 9.2.0.0 fmask 255.255.0.0
```

- Any route that matches 9.2.x.x will be forwarded over the tr1 interface and filtered out on all other interfaces.

Usage Notes

1. The NORECEIVE filter is a replacement for the BLOCK filter.
2. The FORWARD filters are used to control which routes are advertised over which interfaces.
3. The RECEIVE filters are used to control which routes are accepted over which interfaces.
4. NONE is used to clear all filters for a given interface.
5. Care must be taken when using the RIP2B option since host route misinterpretations by adjacent routers running RIP Version 1 can occur.

ROUTED Command

The RouteD command is used to accept the command line parameters listed below. These parameters are valid when starting the program from either the CMS command line or through DTCPARMS.

```
        ┌───────┐
        ↓       │
►►─ROUTED───parms─┴──────────────────────────────────►◄
```

Operands

parms

Any one or more of the following parameters separated by a space. The **[q]** form of an operand deactivates the function associated with that operand.

-f Flush all indirect routes known by RouteD from IP routing tables.

-fh
Flush all indirect host routes known by RouteD from IP routing tables.

-g[q]
Enables the default router. When this option is specified, RouteD will add a default route to its routing information, and broadcast it over all local interfaces. If the adjacent routers add the default route to their routing tables, RouteD 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, you should define a default route to a destination router in the gateways file. See "Configuring a Default Route" on page 174.

Note: Do not use this option if default routes are to be learned dynamically from adjacent routers.

-h[q]
Include host routes, in addition to network routes, for the broadcasts. Adjacent routers should support Host Route Broadcasting to prevent NETWORK UNREACHABLE problems from occurring.

-hv[q]
Include only virtual host routes, in addition to network routes, for the broadcasts. Adjacent routers should support Host Route Broadcasting, or network or subnetwork portions of VIPA addresses must be unique for each TCP/IP image.

-q Suppresses broadcasting of routing information.

-s[q]
Forces RouteD to supply routing information, regardless of whether it is acting as an internetwork router. This is the default if multiple network interfaces are present or if a point-to-point link is used.

-sd[q]
Supplies default route only. When this option is specified, the -g parameter is set internally. This option is provided as a RIP output filter.

-sdv[q]

-svd[q]
Supplies (broadcasts) virtual routes and default routes only. See parameter descriptions for -sv and -sd. This option is provided as a RIP output filter.

-shv[q]

-svh[q]
Supplies (broadcasts) virtual (network and host) routes. This option is provided as a RIP output filter.

-sl[q]
Supplies local (directly-connected) routes only. This option is provided as a RIP output filter.

-st[q]
Supplies triggered updates only. This is similar to the -q parameter, except that RouteD 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[q]
Supplies virtual routes only. Recommended usage is when multiple network adapters in a TCPIP stack are in the same network; otherwise, network connectivity problems will occur. This option is provided as a RIP output filter.

Usage Notes

1. For information on the trace and debug options supported by the RouteD server, see the "Activating RouteD Trace and Debug" section of the *TCP/IP Diagnosis Guide*.
2. Many of these commands and parameters may be enabled or disabled without restarting the RouteD server. See "Using the SMSG Interface to RouteD" on page 179 for information on modifying RouteD operational characteristics.

Configuration Examples

This section contains examples for configuring the RouteD server. The following example illustrates a RouteD configuration.

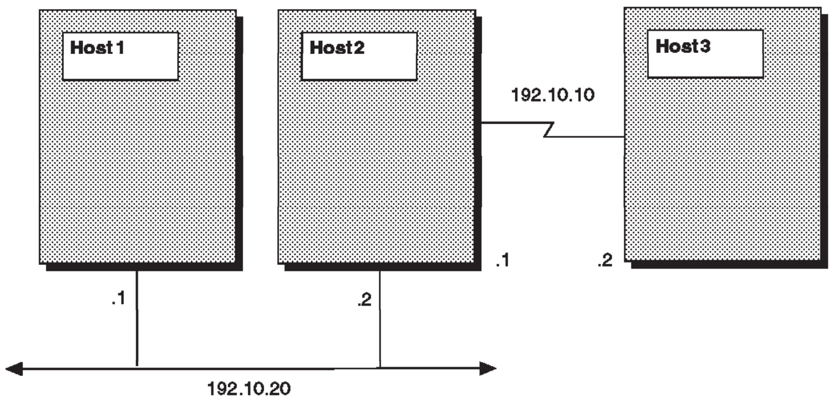


Figure 10. RouteD Configuration Example

Configuring a Passive Route

In Figure 10, assume that your z/VM server is `host1` and is running a RouteD server. The other two hosts, `host2` and `host3`, are not running a RIP server. Your RouteD server does not learn a route to `host3`, because `host2` is not running a RIP server. Your RouteD server sends routing updates to `host3` over the link to `host2` but never receives a routing update from host2. After 180 seconds, your RouteD server deletes the route to host2. This problem is inherent to the RIP protocol and cannot be prevented.

To solve the problem, you should add a passive route to this host in the gateways file. Host1 will maintain this route in its local routing tables indefinitely (no timeout). 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 to it as follows:

```
host  host2       gateway  host1       metric 1  passive
```

A directly-connected passive route is one whose gateway address or name is one of the local interfaces in the HOME list or is one of the offload interfaces.

Assume that your z/VM server is now host2 and is running a RouteD server. host1 is also running a RIP server, but host3 is not. Your RouteD server sends routing information updates to host3 over the link to host3 but never receives a routing update from host3. After 180 seconds, your RouteD 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 10 on page 171, assume that your z/VM server is again host1, which is running a RouteD server. The other two hosts, host2 and host3, are also running RIP servers. Your RouteD 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 RouteD server on host1 from adding a route to host3, add an external route to the gateways file. 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
```

This place holder entry will keep RouteD from adding and propagating the route to this destination host3.

Configuring an Active Gateway

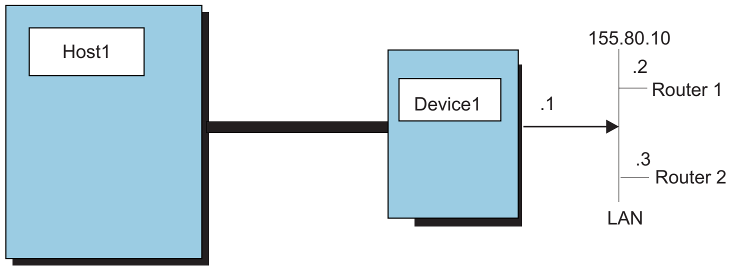


Figure 11. Configuring an Active Gateway

As shown in Figure 11, your z/VM server is `host1`, which is running a RouteD server, and `device1` is a network interface device that does not support link-level broadcasting, or one that does not support ARP processing (for example, HYPERchannel). Also, `host1` is channel-connected to `device1` as a hyperchannel device, and there are routers `router1` and `router2` on the local area network. Because the IP addresses for `router1` and `router2` are unknown by `host1`, they must be manually configured in `host1` for RouteD to communicate with them. Configuring **active gateways** for `router1` and `router2` as remote network interfaces enables RouteD to send RIP updates to them. Include the following definitions in the 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 assignment of a 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
   ```
5. Define an active gateways for the remote routers in the ETC GATEWAYS file.
   ```
   active  active  gateway  155.80.10.2  metric  1  active
   active  active  gateway  155.80.10.3  metric  1  active
   ```

Given these active gateway addresses, RouteD will use them as the destinations for RIP updates to the remote routers. In addition, RouteD will continue to receive RIP updates from the active gateways over the hyperchannel device.

Configuring a Point-to-Point Link

The RouteD 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 server, then passive or permanent routing must be configured in the RouteD ETC GATEWAYS file for the link. See "Configuring a Passive Route" on page 171.

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 and 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 ETC GATEWAYS file. For example, if the default destination router has a gateway address 9.67.112.1, add the following entry to the ETC GATEWAYS file:

```
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. RouteD currently does not support multiple default routes.

Configuring Multiple Subnets with the Same IP Address

It is possible for a network topology to include different internal subnets that each have the same IP subnet address (for example, the two 140.48.96 subnets, Subnet(A) and Subnet(B) in Figure 12 on page 175). This topology can lead to routing ambiguity; however, the RouteD server is able to recognize this topology and dynamically build the routing table entries that are required to direct traffic throughout such a network.

For such a topology, ″network unreachable″ conditions may arise when attempts are made to connect to certain hosts before RouteD has fully initialized its routing tables (which typically requires four to five minutes). For the network shown in Figure 12 on page 175, this might occur if an attempt is made to connect (or simply ping) from Host A to either Host D or Host C, which in either case, requires routing through one of the identically addressed 140.48.96 subnets. However, the unreachable subnet will become reachable once RouteD has initialized its routing tables.

To avoid such problems, define one of the duplicated subnet addresses to have a different subnet address that makes use of variable subnetting. For instance, change Subnet(B) in this sample network to have a subnet address of 140.48.96.64. This will alleviate subnetting ambiguity during RouteD initialization and will ensure that both subnet networks are reachable. Subnetting values for this revised network are listed in Table 13 on page 175.

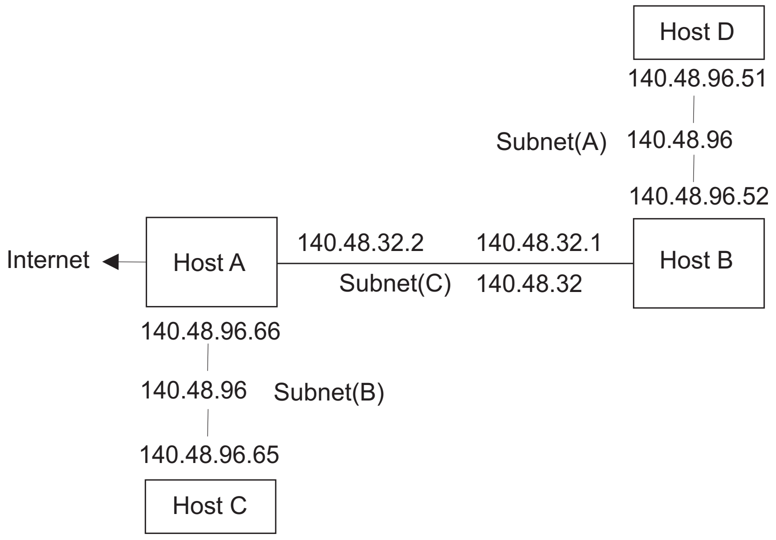


Figure 12. Configuring Multiple Subnets with the Same IP Address

Table 13. Configuring Multiple Subnets with the Same IP Address

HOST	IP ADDRESS	SUBNET MASK	SUBNET ID	HOST ID
HOST A	140.48.32.2	255.255.255.0	140.48.32	2
	140.48.96.66	255.255.255.192	140.48.96.64	2
HOST B	140.48.32.1	255.255.255.0	140.48.32	1
	140.48.96.52	255.255.255.0	140.48.96	52
HOST C	140.48.96.65	255.255.255.192	140.48.96.64	1
HOST D	140.48.96.51	255.255.255.0	140.48.96	51

Representative HOME and BSDROUTINGPARMS statements for Host A's TCP/IP configuration for this (revised) sample network are provided here:

```
HOME
  140.48.32.2    TR2
  140.48.96.66   TR1

BSDROUTINGPARMS FALSE
  ; Link Name MTU   Metric  Subnet Mask     Destination Address
  ; --------- ----- ------- --------------- -------------------
    TR2       2000     0    255.255.255.0          0
    TR1       2000     0    255.255.255.192        0
ENDBSDROUTINGPARMS
```

Configuring RouteD with VIPA

Using RouteD with VIPA gives you the ability to route around interface, device, and network failures.

Assume that you want to configure a VIPA address in a z/VM TCP/IP stack as in Figure 13 on page 177. Here are the configuration steps:

1. Configure the DEVICE, LINK, HOME, ASSORTEDPARMS statements for a VIPA address as described in "Configuring VIPA" on page 39. The assigned VIPA address is 9.1.1.1, and the link name is "VIPA".
2. Update the BSDROUTINGPARMS statement in PROFILE TCPIP for the VIPA link, in addition to the physical links:

```
BSDROUTINGPARMS true
   .
   .
  VIPA   DEFAULTSIZE   0  255.255.255.252 0
   .
   .
ENDBSDROUTINGPARMS
```

Note: the subnet mask is associated with the VIPA link. The subnet mask must be assigned such that the VIPA interface is networked, subnetted, or supernetted. See the subnet mask parameter in "BSDROUTINGPARMS Statement" on page 59.

For more information on configuration options with VIPA, see the following topics:
- "Virtual IP Addressing (VIPA)" on page 37
- "Configuring VIPA" on page 39
- "Using VIPA to Backup or Restore a TCP/IP Stack" on page 40
- "Configuring RouteD to Split Traffic with VIPA"

Configuring RouteD to Split Traffic with VIPA

The purpose of splitting traffic is to reduce traffic load on network interfaces by controlling the inbound and outbound traffic. The following techniques can be used to produce traffic splitting effects with fault tolerance benefits:

- Using Interface Metric and VIPA To Split Inbound/Outbound Traffic

  When there are multiple network interfaces to the same network configuration, split inbound/outbound traffic can be achieved by setting the metric on the primary interface one higher than on the secondary interface(s). From routing updates, an adjacent router uses the gateway of a secondary interface to reach the destination VIPA on the z/VM server, because the route to the gateway has a smaller 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 "BSDROUTINGPARMS Statement" on page 59. An OBEYFILE command can be used for the BSDROUTINGPARMS statement to update an interface metric for a link. For an example of configuring a virtual device, see "Configuring RouteD with VIPA" on page 175.

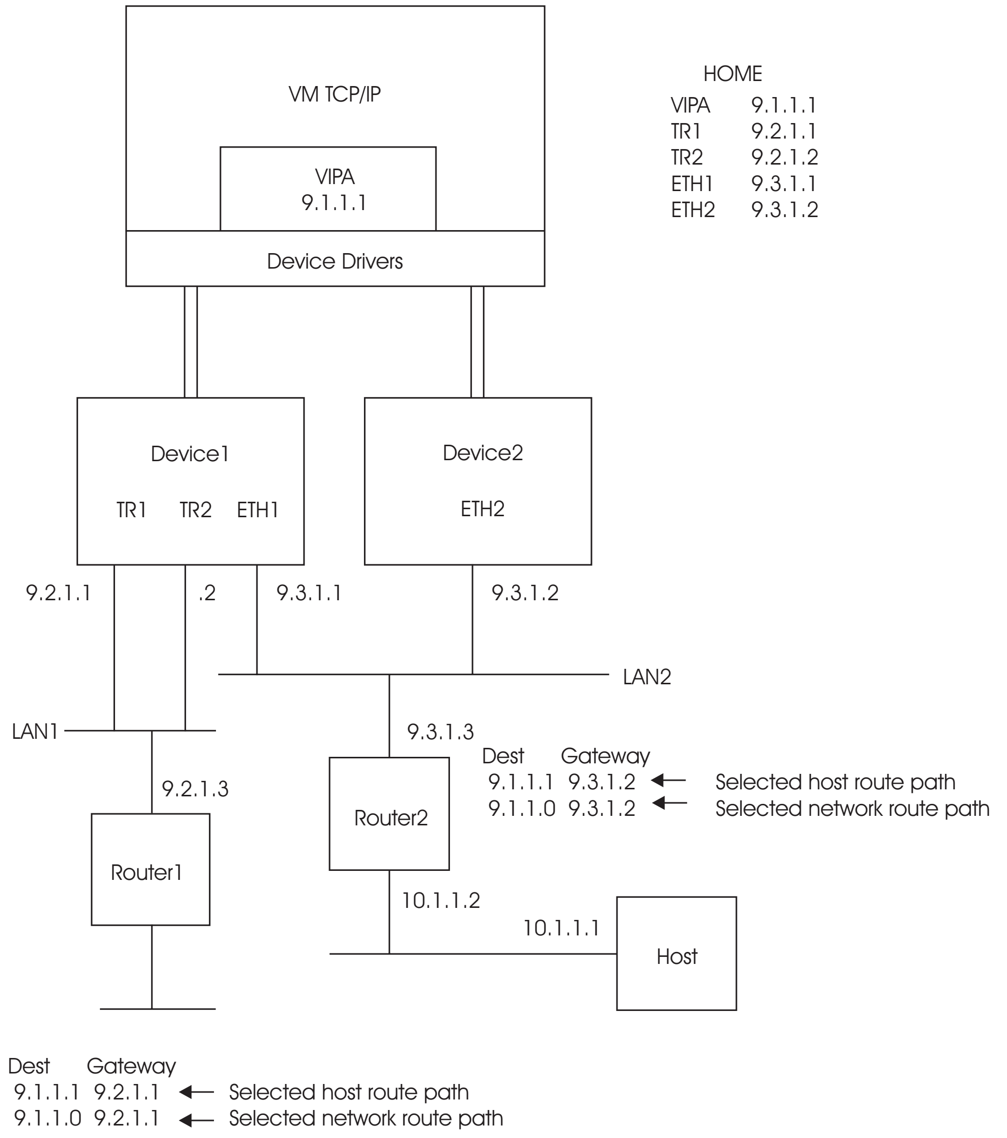


Figure 13. Single VIPA Configuration. Sample configuration showing primary/multiple network interfaces to the same LAN, VIPAs, and inbound/outbound traffic splitting.

ROUTED Server

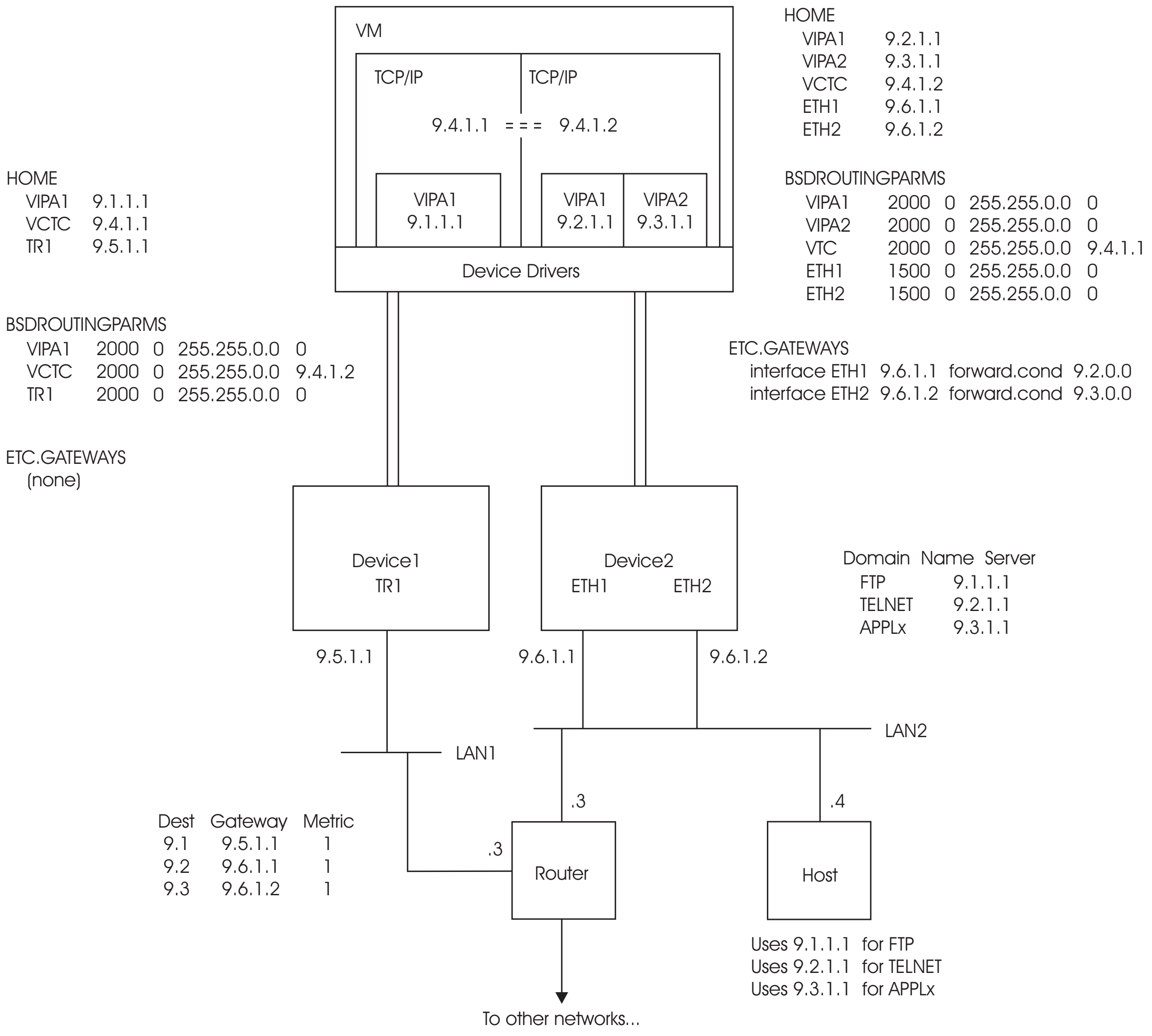


Figure 14. Multiple VIPA Configuration. Sample configuration showing primary/multiple network interfaces to the same LAN, VIPAs and inbound/outbound traffic splitting.

- Using Route Forwarding and VIPA to Split Session Traffic

  With multiple VIPAs in one TCP/IP stack as shown in Figure 14, 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 RouteD. From routing updates, an adjacent router will have multiple gateways to reach the VIPAs on the z/VM 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 z/VM 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 5: Configure the ETC GATEWAYS File (Optional)" on page 162. For an example of configuring a virtual device, see "Configuring RouteD with VIPA" on page 175.

Configuring a Backup TCP/IP Stack with VIPA

To configure an backup z/VM TCP/IP stack with the primary z/VM TCP/IP stack's VIPA address, do the following after a primary z/VM TCP/IP stack is down:

- If RouteD is running on the backup z/VM TCP/IP stack, issue an OBEYFILE command to:
  1. Add a new VIPA device and link.
  2. Add new HOME and BSDROUTINGPARMS statements for the new VIPA link. The HOME statement will have the primary z/VM TCP/IP stack's VIPA address. Ensure that the HOME and BSDROUTINGPARMS statements are included in the same OBEYFILE command.
- If RouteD is not running on the backup z/VM TCP/IP stack, start it and add the new VIPA device and link, HOME and BSDROUTINGPARMS statements using the OBEYFILE command.

Restoring a Primary TCP/IP Stack with VIPA

To restore a VIPA address to the primary z/VM TCP/IP stack after it is no longer needed on the backup, do the following:

- If RouteD is running on the backup z/VM TCP/IP stack, issue an OBEYFILE command to add a new HOME statement with the primary z/VM TCP/IP stack's VIPA address removed. Otherwise, issue an OBEYFILE command to add a new HOME statement with the primary VIPA address removed, and start RouteD. It is not necessary to add a new BSDROUTINGPARMS statement.
- Start RouteD on the primary z/VM 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 z/VM TCP/IP stack.

Using the SMSG Interface to RouteD

The VM Special Message Facility (SMSG) command provides an interface for authorized users to modify RouteD operational characteristics, display routing information, and obtain diagnostic information.

- Authorized users are defined in the OBEY list. See the OBEY statement in "Chapter 4. Configuring the TCP/IP Server" on page 33 for more information on defining authorized users.
- For more information on obtaining diagnostic information, see the section "Activating RouteD Trace and Debug" section in the *TCP/IP Diagnosis Guide*.

SMSG command responses are returned to the originator of the command.

RouteD SMSG Command

Use the SMSG command to enable or disable RouteD command parameters that may or may not have been specified at server initialization or on a previous SMSG command.

```
►►──SMSG──server_id──┬─HELP───────────────────┬──────────────────────────────►◄
                     │         ┌───────┐      │
                     │         ▼       │      │
                     ├─PARMS───┴─parms─┴──────┤
                     ├─CONFIG─────────────────┤
                     ├─RECONFIG───────────────┤
                     ├─RGATEWAYS──────────────┤
                     ├─DGATEWAYS──────────────┤
                     └─SHUTDOWN───────────────┘
```

Operands

server_id
Specifies the user ID of the virtual machine running the VM RouteD server.

HELP
Provides a list of valid SMSG commands accepted by RouteD.

PARMS
Followed by one or more parameters separated by a space. The **[q]** form of an operand deactivates the function associated with that operand.

-f Flush all indirect routes known by RouteD from IP routing tables.

-fh
Flush all indirect host routes known by RouteD from IP routing tables.

-g[q]
Enables the default router. When this option is specified, RouteD will add a default route to its routing information, and broadcast it over all local interfaces. If the adjacent routers add the default route to their routing tables, RouteD 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, you should define a default route to a destination router in the gateways file. See "Configuring a Default Route" on page 174.

Note: Do not use this option if default routes are to be learned dynamically from adjacent routers.

-h[q]
Include host routes, in addition to network routes, for the broadcasts. Adjacent routers should support Host Route Broadcasting to prevent NETWORK UNREACHABLE problems from occurring.

-hv[q]
Include only virtual host routes, in addition to network routes, for the broadcasts. Adjacent routers should support Host Route Broadcasting, or network or subnetwork portions of VIPA addresses must be unique for each TCP/IP image.

-q Suppresses broadcasting of routing information.

-s[q]
Forces RouteD to supply routing information, regardless of whether it is acting as an internetwork router. This is the default if multiple network interfaces are present or if a point-to-point link is used.

-sd[q]
Supplies default route only. When this option is specified, the -g parameter is set internally. This option is provided as a RIP output filter.

-sdv[q]

-svd[q]
Supplies (broadcasts) virtual routes and default routes only. See parameter descriptions for -sv and -sd. This option is provided as a RIP output filter.

-shv[q]

-svh[q]
Supplies (broadcasts) virtual (network and host) routes. This option is provided as a RIP output filter.

-sl[q]
Supplies local (directly-connected) routes only. This option is provided as a RIP output filter.

-st[q]
Supplies triggered updates only. This is similar to the -q parameter, except that RouteD 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[q]
Supplies virtual routes only. Recommended usage is when multiple network adapters in a TCPIP stack are in the same network; otherwise, network connectivity problems will occur. This option is provided as a RIP output filter.

| **CONFIG**
| Reread the RouteD configuration file.

| **RECONFIG**
| Performs dynamic reconfiguration of interfaces and local routes when the
| interface parameters have changed after an OBEYFILE command has been
| issued for a file containing modified HOME and BSDROUTINGPARMS
| statements. Issue this command for RouteD after an OBEYFILE command has
| been issued to reflect dynamic changes made.

| **RGATEWAYS**
| Reread the ETC GATEWAYS file.

| **DGATEWAYS**
| Reread the ETC GATEWAYS file and delete all routes listed.

SHUTDOWN
Stops the RouteD server that is running in the target virtual machine.

Examples

1. The following SMSG command passes parameters to a RouteD server running in the ROUTED2 virtual machine.

```
smsg routed2 parms -h -s -sl
Ready;
09:02:26  * MSG FROM ROUTED2 : PARMS -H -S -SL
```

2. Using the SHUTDOWN SMSG command from another virtual machine to stop the RouteD server running in the ROUTED2 virtual machine.

```
smsg routed2 shutdown
Ready;
14:47:12  * MSG FROM ROUTED2 : SHUTDOWN
```

3. Using the SHUTDOWN SMSG command to stop the RouteD server from the actual server (ROUTED2) virtual machine.

```
CP SMSG * SHUTDOWN
DTCRTD6020I SMSG Command: RouteD Server termination initiated
15:39:04  * MSG FROM ROUTED2 : SHUTDOWN
DTCRUN1014I Server ended normally at 15:39:04 on 1 Apr 1999 (Thursday)
```

Chapter 8. Configuring the MPROUTE Server

This Chapter describes how to configure the Multi-Path Routing (MPROUTE) server. Before presenting the details of the configuration process it is important to explain MPROUTE'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 MPROUTE

MPROUTE 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 VM TCP/IP host running with MPROUTE becomes an active OSPF and/or RIP router in a TCP/IP network. Either (or both) of these routing protocols can be used to dynamically maintain the host routing table. For example, MPROUTE can determine that a new route has been created, that a route is temporarily unavailable, or that 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.

MPROUTE has the following characteristics:

- A one-to-one relationship exists between an instance of MPROUTE and a TCP/IP stack.
- MPROUTE and RouteD cannot run on the same stack concurrently.
- All dynamic routes are deleted from the routing table upon initialization of MPROUTE.
- ICMP Redirects are ignored when MPROUTE is active.
- Unlike RouteD, MPROUTE does not make use of the **BsdRoutingParms** configuration statement. 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 MPROUTE configuration file. Any **BsdRountingParms** that have been specified in the TCPIP configuration file will be overwritten by MPROUTE.
- MPROUTE uses the virtual machine console for its logging and tracing.
- If OSPF and/or RIP is to be used on a particular interface, then the **OSPF_Interface** or **RIP_Interface** configuration statement must be configured.
- Interfaces which are not to be involved in the communication of the RIP or OSPF protocol (such as VIPA interfaces) must be configured to MPROUTE 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).
- MPROUTE supports Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. The virtual routes are included in the OSPF and RIP advertisements to adjacent routers. Adjacent routers learn about virtual routes from the advertisements and can use them to reach the destinations at the VM host.

Use of static routes (defined via the GATEWAY TCP/IP configuration statement) along with MPROUTE 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.

MPROUTE works best without static routes. If, however, 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 MPROUTE. Set **IMPORT_STATIC_ROUTES** to YES on the **AS_Boundary Routing** configuration statement or set **SEND_STATIC_ROUTES** to YES on the **RIP_Interface** configuration statement if you wish for the static routes to be advertised to other routers.

For details on the statements in the MPROUTE configuration file, see the "OSPF Configuration Statements" on page 190.

Various commands can be issued to the MPROUTE server by using the SMSG interface. For details on MPROUTE's SMSG interface see "SMSG Interface to the MPROUTE Server" on page 190.

Open Shortest Path First (OSPF)

OSPF is classified as an Interior Gateway Protocol (IGP) that distributes routing information within a single Autonomous System (AS). The OSPF protocol is a routing protocol designed expressly for the TCP/IP internet environment, including explicit support for IP subnetting and the tagging of externally-derived routing information.

OSPF 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 routes during 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 (e.g., the router's usable interfaces and reachable neighbors). The router distributes its local state throughout the Autonomous System by propagation.

From the topological database, each OSPF 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. The metric is a non–negative integer, where the lower the number, the greater the preference.

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 other areas in 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 (i.e., different masks). This is commonly referred to as variable length subnetting. A packet is routed to the best (i.e., longest or most specific) match. Host routes are considered to be subnets whose masks are ″all ones″.

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 (e.g., 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 an IGP 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 MPROUTE, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. See "Appendix B. Related Protocol Specifications" on page 623 for more information about how you can obtain information on RFCs.

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 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. MPROUTE 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. MPROUTE will not generate immediate next hops, but will preserve them if they are included in the RIP packets.
- Multicasting to reduce load on hosts

  IP multicast address 224.0.0.9 is reserved for RIP Version 2 packets. This is used to reduce unnecessary processing of packets by 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 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.

Note: When sending and receiving Rip packets, either Rip Version 1 or Rip Version 2 can be enabled on an interface; both versions cannot be enabled at the same time.

Configuration Steps for the MPROUTE Server

MPROUTE Server Configuration Steps

1. Update the TCPIP server configuration file, (PROFILE TCPIP).
2. Update the ETC SERVICES file.
3. Create the MPROUTE configuration file, (MPROUTE CONFIG).
4. Update the DTCPARMS file for the MPROUTE server (Optional).

Dynamic Server Operation

The MPROUTE server provides a VM Special Message (SMSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information see “SMSG Interface to the MPROUTE Server” on page 190.

Step 1: Update PROFILE TCPIP

The name of the MPROUTE server must be included in the OBEY Statement because of its dependence on the Raw Sockets. In the ASSORTEDPARMS Statement, you should insure that the **IGNOREREDIRECT** as well as the

VARSUBNETTING Operands are specified. Also include the MPROUTE server virtual machine user ID in the AUTOLOG statement of the TCPIP server configuration file. The MPROUTE server is then automatically started when TCPIP is initialized. The IBM default user ID for this server is **MPROUTE**. Verify that the following statement has been added to the PROFILE TCPIP file:

```
AUTOLOG
  MPROUTE  0
```

If the RIP protocol of MPROUTE is going to be used, UDP port 520 should be reserved for MPROUTE. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
      520 UDP MPROUTE   ;   MPROUTE Server
```

Autolog Considerations for MPROUTE

If a server in the AUTOLOG list also has a PORT Statement reserving a TCP or UDP port but does not have a listening connection on that port, TCP/IP will periodically attempt to cancel and then restart that server. Therefore, if MPROUTE is being started with AUTOLOG and only the OSPF protocol is being used (no RIP protocol, and therefore no listening connection on the RIP UDP port), you **must** do one of the following:

- Ensure that the RIP UDP port (520) is not reserved by the PORT statement in the TCPIP server configuration file.
- Add the NOAUTOLOG parameter to the PORT statement in the TCPIP server configuration file. For example,

  ```
  PORT
  520 UDP MPROUTE NOAUTOLOG
  ```

Note: When using only the OSPF protocol, the auto-start feature of AUTOLOG can be used as described above. However, the monitoring and auto-restart features of AUTOLOG are unavailable due to AUTOLOG’s dependence on a listening TCP or UDP connection, which does not exist with OSPF.

If you fail to take one of the above actions, MPROUTE will be periodically cancelled and restarted by TCP/IP.

Step 2: Update the ETC SERVICES file

The ETC SERVICES file contains the relationship between services and port numbers. If the RIP protocol of MPROUTE is going to be used, ensure that the following line is added to the ETC SERVICES file:

```
router       520/udp     route MPROUTE
```

Note: In the above example, the default well–known port (520) is used.

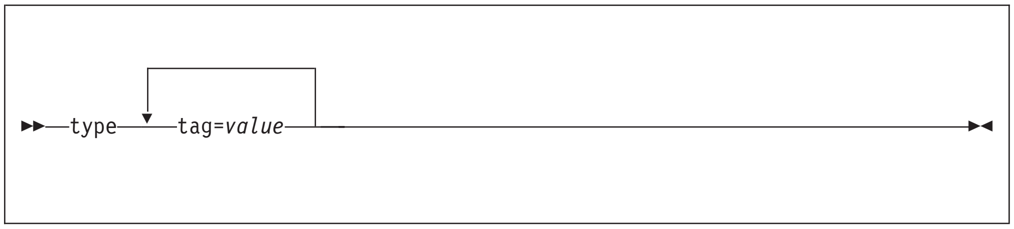
You should compare the port number configured in the ETC SERVICES file to the port number configured for the MPROUTE server in the PROFILE TCPIP file to ensure that the port numbers are the same.

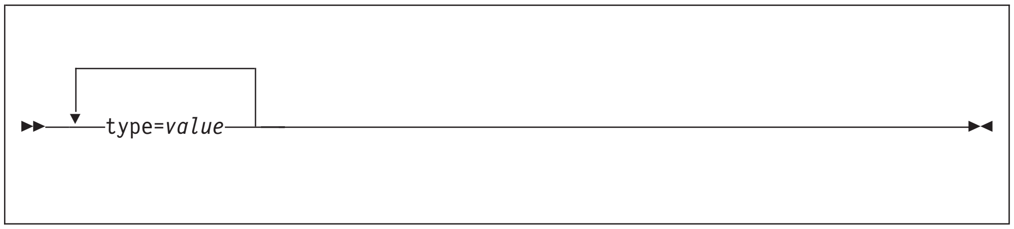
Step 3: Create the MPROUTE Configuration File

The MPROUTE configuration file contains the OSPF and/or RIP configuration definitions. Please refer to the complete list of allowable OSPF and/or RIP configuration statements (“OSPF Configuration Statements” on page 190) when preparing the MPROUTE configuration file. A sample of this configuration file is shipped as MPROUTE SCONFIG on TCPMAINT 591. It should be copied to TCPMAINT 198, customized, and renamed to MPROUTE CONFIG.

Syntax Rules

Statements in the MPROUTE configuration file have the following syntax:





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 MPROUTE configuration statements:

- Types, tags, and values can be specified in mixed case.
- Every configuration statement must end with a semicolon.
- Blanks and comments are supported. Comments are identified by a semicolon in any column. Comments cannot appear within a configuration statement.
- Statements may begin in any column.

Step 4: Update the DTCPARMS File (Optional)

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

```
:nick.MPROUTE
  :PARMS.
```

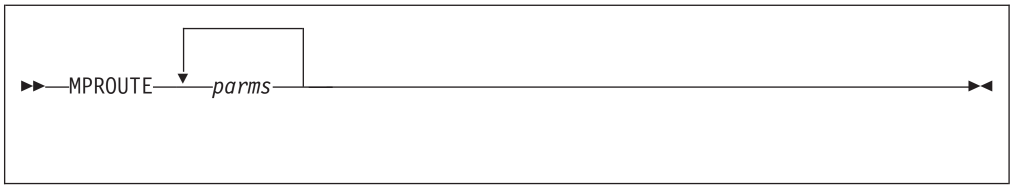
If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see “Chapter 3. General TCP/IP Server Configuration” on page 17.

Note: You should modify the DTCPARMS file for the MPROUTE server if you need to specify any of the MPROUTE server input parameters. See the “MPROUTE Command” for a complete list.

MPROUTE Command

The MPROUTE command is used to accept the command line parameters listed below. These parameters are valid when starting the program from either the CMS command line or through DTCPARMS.



Operands

parms
Any one or more of the following parameters separated by a space.

-tn (where n is a supported trace level)
This option specifies the external tracing level. It is intended for general use 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
```

-dn (where n is a supported debug level)
This option specifies the internal debugging level. It is intended for service or developers, and provides internal debugging information needed for debugging problems. The following levels are supported:

```
1 = Internal debugging messages
2 = Unformatted hex packet trace
3 = Function entry/exit trace
4 = Task add/run
```

For **–t** and **–d** the above option levels are treated as cumulative; that is, level 2 includes level 1. For example, -t2 provides formatted packet trace and informational messages.

Usage Notes

1. For more information on the trace and troubleshooting problems with the MPROUTE server, see the “Activating MPROUTE Trace and Debug” section of the *TCP/IP Diagnosis Guide*.

Dynamic Server Operation

The VM Special Message Facility (SMSG) command provides an interactive interface to the MPROUTE virtual machine to perform privileged system administration tasks, such as:
- shutting down the server
- changing the cost of the interface
- displaying valid SMSG command options

Privileged users are specified in the OBEY list of the TCPIP server configuration file.

Note: Command responses are returned to the originator of the command through CP MSG commands.

SMSG Interface to the MPROUTE Server

```
►►──SMSG──server_id──┬─HELP──────────────────────────────┬──────────────►◄
                     ├─OSPF──WEIGHT────NAME=name───COST=cost─┤
                     └─SHUTDOWN──────────────────────────┘
```

Operands

server_id
Specifies the user ID of the MPROUTE server virtual machine.

HELP
Provides a list of valid SMSG commands accepted by MPROUTE.

OSPF WEIGHT NAME=*name* **COST=***cost*
The cost of an OSPF interface can be dynamically changed using the NAME/COST operand. The new cost is propagated 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 MPROUTE configuration file.

SHUTDOWN
Stops the MPROUTE server that is running in the target virtual machine. The **#CP EXTERNAL** command cannot be used for shutdown.

1. An example using the SHUTDOWN SMSG command from another virtual machine to stop the MPROUTE server running in the MPROUTE1 virtual machine follows:

```
smsg mproute1 shutdown
Ready;
12:37:10  * MSG FROM MPROUTE1 : SHUTDOWN
```

2. This example is using the SHUTDOWN SMSG command to stop the MPROUTE server from the actual server (MPROUTE1) virtual machine:

```
CP SMSG * SHUTDOWN
DTCRTD6020I SMSG Command: MPROUTED Server termination initiated
13:39:02  * MSG FROM MPROUTED1 : SHUTDOWN
DTCRUN1014I Server ended normally at 13:39:04 on 3 Mar 2000 (Friday)
```

For more detailed information on using the SMSG interface for diagnosing and troubleshooting problems with the MPROUTE server, see the *TCP/IP Diagnosis Guide*.

OSPF Configuration Statements

This section contains descriptions of the OSPF configuration statements:
- AREA
- COMPARISON
- OSPF_INTERFACE
- VIRTUAL_LINK
- ROUTERID
- AS_BOUNDARY_ROUTING
- RANGE

- DEMAND_CIRCUIT

Note: Statements with only optional operands must have at least one operand coded, even if all operands have defaults.

AREA Statement

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).

```
            ┌─Area_Number=0.0.0.0──────────────┐
►►──Area────┤                                  ├──────────────────────────────────────►
            └─Area_Number=ospf_area_address────┘

   ┌─Authentication_Type=None─────────────────┐   ┌─Stub_Area=NO──┐
►──┤                                          ├───┤               ├──────────────────►
   └─Authentication_Type=security_scheme──────┘   └─Stub_Area=YES─┘

   ┌─Stub_Default_Cost=1──────┐   ┌─Import_Summaries=YES─┐
►──┤                          ├───┤                      ├───────────────────────────►◄
   └─Stub_Default_Cost=cost───┘   └─Import_Summaries=NO──┘
```

Operands

Area_Number=*ospf_area_address*
Specifies the OSPF area address in dotted decimal.

Area ID 0.0.0.0 is reserved for the backbone. The backbone is responsible for summarizing the Autonomous System of each area to every other area. All inter-area traffic must pass through the backbone; non backbone areas cannot exchange packets directly.

Authentication_Type=*security_scheme*
Specifies 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 or not this area is a stub area. 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=*cost*
Specifies the cost OSPF associates with the default route to its 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 Statement

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.

```
   ┌─Comparison=Type2─┐
►►─┤                  ├──────────────────────────────────────────────►◄
   └─Comparison=Type1─┘
```

Operands

Comparison=Type*x*
Compare to type 1 or 2 externals. Valid values are Type1 (or 1) or Type2 (or 2).

OSPF_INTERFACE Statement

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.

```
►►──OSPF_Interface──IP_address=ip_address──Name=interface_name───────────────►

►─Subnet_mask=subnet_mask──┬──────────────────────────┬──────────────────────►
                           └─Destination_Addr=address─┘

   ┌─Attaches_To_Area=0.0.0.0─┐ ┌─MTU=576──┐
►──┼──────────────────────────┼─┼──────────┼─────────────────────────────────►
   └─Attaches_To_Area=area────┘ └─MTU=size─┘

   ┌─Retransmission_Interval=5─────────┐ ┌─Transmission_Delay=1─────┐
►──┼───────────────────────────────────┼─┼──────────────────────────┼────────►
   └─Retransmission_Interval=frequency─┘ └─Transmission_Delay=delay─┘

   ┌─Router_Priority=1────────┐ ┌─Hello_Interval=10───────┐
►──┼──────────────────────────┼─┼─────────────────────────┼──────────────────►
   └─Router_Priority=priority─┘ └─Hello_Interval=interval─┘

   ┌─Dead_Router_Interval=40───────┐ ┌─Cost0=1────┐ ┌─Subnet=NO──┐
►──┼───────────────────────────────┼─┼────────────┼─┼────────────┼───────────►
   └─Dead_Router_Interval=interval─┘ └─Cost0=cost─┘ └─Subnet=YES─┘

   ┌─Authentication_Key=nulls──────┐ ┌─Demand_Circuit=NO──┐
►──┼───────────────────────────────┼─┼────────────────────┼──────────────────►
   └─Authentication_Key="password"─┘ └─Demand_Circuit=YES─┘

   ┌─Hello_Suppression=Allow─┐ ┌─PP_Poll_Interval=60───────┐
►──┼─────────────────────────┼─┼───────────────────────────┼─────────────────►
   └─Hello_Suppression=value─┘ └─PP_Poll_Interval=interval─┘

   ┌─Parallel_OSPF=Backup─┐ ┌─Non_Broadcast=NO──┐
►──┼──────────────────────┼─┼───────────────────┼────────────────────────────►
   └─Parallel_OSPF=value──┘ └─Non_Broadcast=YES─┘

   ┌─NB_Poll_Interval=120──────┐
►──┼───────────────────────────┼─┬─────────────────────────┬─────────────────►
   └─NB_Poll_Interval=interval─┘ └─DR_Neighbor=ip_address──┘

                                 ┌─Max_Xmit_Time=120──────┐
►──┬───────────────────────────┬─┼────────────────────────┼──────────────────►
   └─No_DR_Neighbor=ip_address─┘ └─Max_Xmit_Time=max_time─┘

   ┌─Min_Xmit_Time=0.5──────┐ ┌─RT_Gain=0.125──┐ ┌─Variance_Gain=0.25───┐
►──┼────────────────────────┼─┼────────────────┼─┼──────────────────────┼────►
   └─Min_Xmit_Time=min_time─┘ └─RT_Gain=rtgain─┘ └─Variance_Gain=vgain──┘

   ┌─Variance_Mult=2────┐ ┌─Delay_Acks=NO──┐
►──┼────────────────────┼─┼────────────────┼─────────────────────────────────►◄
   └─Variance_Mult=mult─┘ └─Delay_Acks=YES─┘
```

Operands

IP_address=*ip_address*
Specifies the IP address of the local interface to be configured for OSPF.

This parameter can be a valid IP address that is configured on the system or it can be specified with asterisks (*) as wild cards. The valid wildcard 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=*interface_name*
Specifies the name of the interface that must match the link name coded for the corresponding IP address on the HOME statement in the TCP/IP profile. This parameter is ignored on wildcard interface definitions.

Subnet_Mask=*subnet_mask*
Specifies the subnet mask of the network this interface attaches to.

Destination_Add=*address*
Specifies the IP address of the host at the remote end of this interface. This parameter is only valid for point-to-point links. If this parameter is not specified for a point-to-point link, a route to the host at the remote end of the interface will not be added to the TCP/IP route table until OSPF communication is established with that host. A subnet route for the interface will be added at MPROUTE initialization independent of whether this parameter is specified.

Attaches_To_Area=*area*
Indicates the OSPF area to which this interface attaches. Valid values are 0.0.0.0 (the backbone), or any area defined by the AREA statement.

MTU=*size*
Specifies the mtu size for OSPF to add to the routing table for routes that take this interface. Valid values are 0 to 65535.

Retransmission_Interval=*frequency*
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=*delay*
Specifies an estimate of the number of seconds it takes to transmit link-state information over the interface. Each link-state advertisement has a 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=*priority*
Specifies the designated router to be used for broadcast and non-broadcast multiaccess networks, 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=*interval*
Indicates how often (in seconds) OSPF will send out over 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=*interval*
: Specifies 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 four times 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=*cost*
: Specifies the shortest path to be used to a destination. Valid values are 1 to 65535.

Subnet
: For an interface to a point-to-point serial line, 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=″*password*″
: Specifies the password for the OSPF routers attached to the network. Coded when Authentication_Type=Password on the AREA statement for the area to which this interface attaches is used. This value must be the same for all routers attached to a common network. Valid values are any 8 characters from code page 1047 coded within double quotes or any hex string which begins with ″0x″.

Demand_Circuit
: Specifies whether the Link State Advertisements (LSAs) transmitted over this interface will expire. 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=*value*
: Specifies the hello_suppression value. Meaningful only if Demand_Circuit is coded YES. 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=*interval*
: Specifies how often (in seconds) OSPF will 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=*value*
: Indicates whether the OSPF interface is primary or backup when more than one OSPF interface is defined to the same subnet. Only one of these interfaces can be configured as primary, meaning that it will be the interface to carry the OSPF protocol traffic between MPROUTE and the subnet. Failure of the primary interface results in automatic switching of OSPF traffic to one of the backup interfaces. If none of the interfaces to the common subnet are configured as primary, a primary interface will be selected by MPROUTE. Valid values are ″Backup″ or ″Primary″.

Non_Broadcast
: If the router is connected to a nonbroadcast, multiaccess (NBMA) network,

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=*interval*
Specifies how often (in seconds) Hello packets are sent to neighbors that are inactive. You must set this poll interval consistently across all interfaces that attach to the same network 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=*ip_address*
Defines which routers within the non-broadcast network can become a designated router. If the OSPF protocol communicates with one or more routers over a non_broadcast network interface, MPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate. For non-broadcast network interfaces, there is no underlying signaling that allows the stack to learn the required IP addresses. As a result, the neighbor addresses must be configured to MPROUTE with this parameter. Multiple **DR_Neighbor** statements may be coded on an **OSPF_interface** statement as necessary. The value of the *ip_address* must be a valid IP address.

No_DR_Neighbor=*ip_address*
Defines which routers within the non-broadcast network cannot become a designated router. If the OSPF protocol communicates with one or more routers over a non_broadcast network interface, MPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate. For non-broadcast network interfaces, there is no underlying signaling that allows the stack to learn the required IP addresses. As a result, the neighbor addresses must be configured to MPROUTE with this parameter. Multiple **No_DR_Neighbor** statements may be coded on an **OSPF_interface** statement as necessary. The value of the *ip_address* must be a valid IP address.

Max_Xmit_Time=*max_time*
Specifies the maximum retransmit time (in seconds) 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 acknowledgment before resending a packet. Valid values are 0 to 999.990 seconds.

Min_Xmit_Time=*min_time*
Specifies the minimum retransmit time (in seconds) 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 acknowledgment before resending a packet. Valid values are 0 to 999.990 seconds.

RT_Gain=*rtgain*
Indicates 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 acknowledgment before resending a packet. Valid values are 0 to 1.0.

Variance_Gain=*vgain*
: Indicates 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 acknowledgment before resending a packet. Valid values are 0 to 1.0.

Variance_Mult=*mult*
: Indicates 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 acknowledgment before resending a packet. Valid values are 0 to 99.990.

Delay_Acks
: Specifies the delay acknowledgments value to add to the routing table for routes that take this interface. Specifying YES delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specifying NO results in acknowledgments being returned immediately. Valid values are YES and NO.

VIRTUAL_LINK Statement

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 links which are part of Virtual IP Address support (VIPA).

```
►►──Virtual_Link──Virtual_Endpoint_RouterID=id─────────────────────────────────►

   ┌─Links_Transit_Area=0.0.0.1─┐  ┌─Retransmission_Interval=10────────┐
►──┴─Links_Transit_Area=area────┴──┴─Retransmission_Interval=frequency─┴───────►

   ┌─Transmission_Delay=5─────┐  ┌─Hello_Interval=30───────┐
►──┴─Transmission_Delay=delay─┴──┴─Hello_Interval=interval─┴───────────────────►

   ┌─Dead_Router_Interval=180──────┐  ┌─Authentication_Key=nulls───────┐
►──┴─Dead_Router_Interval=interval─┴──┴─Authentication_Key="password"──┴───────►◄
```

Operands

Virtual_Endpoint_RouterID=*id*
: Indicates the router ID of the virtual neighbor (other endpoint). Router IDs are entered in the same form as IP addresses.

Links_Transit_Area=*area*
: Specifies 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=*frequency*
Specifies how often (in seconds) of retransmitting the 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=*delay*
Specifies 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=*interval*
Specifies 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=*interval*
Specifies 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=″*password*″
Specifies the password for the virtual links transit area. Valid values are any 8 characters from code page 1047 coded within double quotes or any hex string that begins with ″0x″.

ROUTERID Statement

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 network addresses.
- If the RouterID is not configured, one of the OSPF interface addresses will be used as the OSPF router ID.

```
►►──RouterID=id──────────────────────────────────────────────────►◄
```

Operands

RouterID *id*
Specifies the ID, which is one of the TCP/IP stack's configured interface IP network addresses, in dotted decimal form.

AS_BOUNDARY_ROUTING Statement

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.

```
                           ┌─Import_RIP_Routes=NO──┐
►►──AS_Boundary_Routing────┤                       ├───────────────────────────────►
                           └─Import_RIP_Routes=YES─┘

  ┌─Import_Static_Routes=NO──┐  ┌─Import_Direct_Routes=NO──┐
►─┤                          ├──┤                          ├──────────────────────►
  └─Import_Static_Routes=YES─┘  └─Import_Direct_Routes=YES─┘

  ┌─Import_Subnet_Routes=YES─┐  ┌─Originate_Default_Route=NO──┐
►─┤                          ├──┤                             ├───────────────────►
  └─Import_Subnet_Routes=NO──┘  └─Originate_Default_Route=YES─┘

  ┌─Originate_as_Type=2─┐  ┌─Default_Route_Cost=1────┐
►─┤                     ├──┤                         ├────────────────────────────►
  └─Originate_as_Type=1─┘  └─Default_Route_Cost=cost─┘

►─┬─────────────────────────────────────┬─────────────────────────────────────────►◄
  └─Default_Forwarding_Address=address──┘
```

Operands

Import_RIP_Routes
Specifies whether RIP routes 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 a default router. Valid values are YES or NO.

Originate_as_Type
Specifies the external type assigned to the default route. Valid values are 1 or 2.

Default_Route_Cost=*cost*
Specifies the cost that OSPF associates with the default route. Valid values are 0 to 16777215.

Default_Forwarding_Address=*address*
Specifies the forwarding address that will be used in the imported default route.

RANGE Statement

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.

```
                                                  ┌─Area_Number=0.0.0.0─┐
►►──Range──IP_address=address──Subnet_Mask=mask───┼─────────────────────┼──────►
                                                  └─Area_Number=area────┘

   ┌─Advertise=YES─┐
►──┼───────────────┼───────────────────────────────────────────────────────────►◄
   └─Advertise=NO──┘
```

Operands

IP_Address=*address*
Specifies common subnet portion of IP addresses in this range. Valid values are valid IP network addresses in dotted decimal form.

Subnet_Mask=*mask*
Specifies the subnet mask with respect to the network range defined in IP_Address.

Area_Number=*area*
Specifies the area number for which to add this range. Valid values are any defined areas in dotted decimal form.

Advertise
Specifies whether this range will be advertised to other areas. Valid values are YES or NO.

DEMAND_CIRCUIT Statement

Demand circuits are a usage-sensitive connection, such as the X.25 protocol. Coding this global statement with a YES enables demand circuits. Demand circuit parameters can then be coded on the **OSPF_Interface statement**.

```
   ┌Demand_Circuit=YES┐
►►─┤                  ├──────────────────────────────────────────────►◄
   └Demand_Circuit=NO─┘
```

Operands

Demand_Circuit
Valid values are YES or NO.

RIP Configuration Statements

This section contains descriptions of the RIP configuration statements:
- ORIGINATE_RIP_DEFAULT
- RIP_INTERFACE
- ACCEPT_RIP_ROUTE

ORIGINATE_RIP_DEFAULT Statement

Indicates under what conditions RIP will support Default route (destination/mask 0.0.0.0/0.0.0.0) generation.

```
                             ┌Condition=Always────┐  ┌Cost=1────┐
►►──Originate_RIP_Default────┤                    ├──┤          ├──────►◄
                             └Condition=condition─┘  └Cost=cost─┘
```

Operands

Condition=*condition*
Indicates the condition for when RIP is to advertise this route as a default route. Valid values are:
- Always: Always originate RIP default.
- OSPF: Originate RIP default if OSPF routes are available.

- Never: Never advertise this route as a default route.

Cost=*cost*
Specifies the cost that RIP will advertise with the default route that it originates. Valid values are 1 to 16.

RIP_INTERFACE Statement

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.

```
►►──RIP_Interface──IP_address=ip_address──Name=interface_name──────────────►

                                                      ┌─MTU=576──┐
►─Subnet_mask=subnet_mask──┬──────────────────────────┬┼──────────┼──────────►
                           └─Destination_Addr=address─┘└─MTU=size─┘

  ┌─Receive_RIP=YES─┐  ┌─Receive_Dynamic_Nets=YES─┐
►─┼─────────────────┼──┼──────────────────────────┼────────────────────────────►
  └─Receive_RIP=NO──┘  └─Receive_Dynamic_Nets=NO──┘

  ┌─Receive_Dynamic_Subnets=YES─┐  ┌─Receive_Dynamic_Hosts=NO──┐
►─┼─────────────────────────────┼──┼───────────────────────────┼──────────────►
  └─Receive_Dynamic_Subnets=NO──┘  └─Receive_Dynamic_Hosts=YES─┘

  ┌─Send_RIP=YES─┐  ┌─Send_Default_Routes=NO──┐  ┌─Send_Net_Routes=YES─┐
►─┼──────────────┼──┼─────────────────────────┼──┼─────────────────────┼──────►
  └─Send_RIP=NO──┘  └─Send_Default_Routes=YES─┘  └─Send_Net_Routes=NO──┘

  ┌─Send_Subnet_Routes=YES─┐  ┌─Send_Static_Routes=NO──┐
►─┼────────────────────────┼──┼────────────────────────┼──────────────────────►
  └─Send_Subnet_Routes=NO──┘  └─Send_Static_Routes=YES─┘

  ┌─Send_Host_Routes=NO──┐  ┌─Send_Poisoned_Reverse_Routes=YES─┐
►─┼──────────────────────┼──┼──────────────────────────────────┼──────────────►
  └─Send_Host_Routes=YES─┘  └─Send_Poisoned_Reverse_Routes=NO──┘

  ┌─In_Metric=1──────┐
►─┼──────────────────┼─────────────────────────────────────────────────────────►
  └─In_Metric=metric─┘

  ┌─Out_Metric=0──────┐  ┌─RipV2=NO──┐  ┌─RipV1_Routes=NO──┐
►─┼───────────────────┼──┼───────────┼──┼──────────────────┼──────────────────►
  └─Out_Metric=metric─┘  └─RipV2=YES─┘  └─RipV1_Routes=YES─┘

                                    ┌──────────────────────────┐
  ┌─Authentication_Key=nulls─┐      ▼                          │
►─┼──────────────────────────┼──────┴──┬──────────────────────┬┴──────────────►
  └─Authentication_Key=key───┘         └─Neighbor=ip_address──┘

  ┌─Max_Xmit_Time=120──────┐  ┌─Min_Xmit_Time=0.5──────┐
►─┼────────────────────────┼──┼────────────────────────┼──────────────────────►
  └─Max_Xmit_Time=max_time─┘  └─Min_Xmit_Time=min_time─┘

  ┌─RT_Gain=0.125──┐  ┌─Variance_Gain=0.25──┐  ┌─Variance_Mult=2────┐
►─┼────────────────┼──┼─────────────────────┼──┼────────────────────┼─────────►
  └─RT_Gain=rtgain─┘  └─Variance_Gain=vgain─┘  └─Variance_Mult=mult─┘

  ┌─Delay_Acks=NO──┐
►─┼────────────────┼─────────────────────────────────────────────────────────►◄
  └─Delay_Acks=YES─┘
```

Operands

IP_address=*ip_address*
Specifies the 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 wildcard 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=*interface name*
Specifies the name of the interface. This must match the link name coded for the corresponding IP address on the HOME statement in the TCP/IP profile. This parameter is ignored on wildcard interface definitions.

Subnet_Mask=*mask*
Specifies the subnet mask for the associated interface IP address in dotted decimal form.

Destination_Addr=*address*
Specifies the IP address in dotted decimal form 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 MPROUTE initialization independent of whether this parameter is specified.

MTU=*size*
Specifies the mtu size for RIP to add to the routing table for routes that take this interface. Valid values are 0 to 65535.

Receive_RIP
Specifies whether 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 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 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 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.

Send_RIP
Specifies whether RIP advertisements will be broadcast over this interface. Valid values are YES or NO.

Send_Default_Routes
Specifies whether the default route (destination 0.0.0.0), if it is available in RIP responses, will be sent from this IP source address. Valid values are YES or NO.

Send_Net_Routes
Specifies whether all network level routes in RIP responses will be sent from this IP address. Valid values are YES or NO.

Send_Subnet_Routes
Specifies whether appropriate subnet-level routes in RIP responses will be 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
Specifies whether static and direct routes in RIP responses will be sent from this IP source address. Valid values are YES or NO.

Send_Host_Routes
Specifies whether host routes in RIP responses will be 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
Specifies whether poisoned reverse routes over the interface will be corresponding to the next hop. A poison reverse route is one with an infinite metric (i.e. 16). Valid values are YES or NO.

In_Metric=*inmetric*
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=*outmetric*
Specifies the value of the metric to be added to RIP routes advertised over this interface. Valid values are 0 to 15.

RipV2
Specifies whether RIP V2 will be enabled on this link. Valid values are YES or NO.

RipV1_Routes
Specifies whether RIP V1 routes will be advertised on this RIP V2 link. Valid values are YES or NO.

Authentication_Key=*key*
Specifies the 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″.

Neighbor=*ip_address*
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. The value can be any valid IP address.

Max_Xmit_Time=*max_time*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 999.990 seconds.

Min_Xmit_Time=*min_time*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 99.990 seconds.

RT_Gain=*rtgain*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 1.0.

Variance_Gain=*vgain*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 1.0.

Variance_Mult=*mult*
Specifies 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 acknowledgment 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 acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specifying NO results in acknowledgments being returned immediately. Valid values are YES and NO.

ACCEPT_RIP_ROUTE Statement

Allows a RIP network, subnet, or host route to be accepted independently 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.

```
►►──Accept_RIP_Route──IP_address=ip_address──────────────────────────────►◄
```

Operands

IP_address=*ip_address*
Specifies the destination route to be unconditionally accepted.

Common Configuration Statements for RIP and OSPF

This section contains descriptions of the Common configuration statements:
- DEFAULT_ROUTE
- INTERFACE

DEFAULT_ROUTE Statement

Allows the default route to be specified to MPROUTE. 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.

```
►►──Default_Route──Name=interface_name──Next_Hop=ip_address──────────────►◄
```

Operands

Name=*interface name*
: Specifies 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*
: Specifies the IP address of the next hop.

INTERFACE Statement

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 MPROUTE via the INTERFACE statement, unless it is a non-point-to-point interface and the default values for for Subnet Mask and MTU are acceptable for that interface. The default value for MTU is 576. The following table lists the defaults for the subnet Mask:

Class Address	First Octet	Default Subnet Mask
A	(0-127)	255.0.0.0
B	(128-191)	255.255.0.0
C	(192-223)	255.255.255.0

```
►►──Interface──IP_address=ip_address──Name=interface_name──Subnet_Mask=mask────►

                                      ┌─MTU=576──┐  ┌─Max_Xmit_Time=120──────┐
►──┬─────────────────────────────┬──┼──────────┼──┼────────────────────────┼──►
   └─Destination_Addr=address─┘  └─MTU=size─┘  └─Max_Xmit_Time=max_time─┘

   ┌─Min_Xmit_Time=0.5──────┐  ┌─RT_Gain=0.125─┐  ┌─Variance_Gain=0.25──┐
►──┼────────────────────────┼──┼───────────────┼──┼─────────────────────┼─────►
   └─Min_Xmit_Time=min_time─┘  └─RT_Gain=rtgain─┘  └─Variance_Gain=vgain─┘

   ┌─Variance_Mult=2────┐  ┌─Delay_Acks=NO──┐
►──┼────────────────────┼──┼────────────────┼─────────────────────────────►◄
   └─Variance_Mult=mult─┘  └─Delay_Acks=YES─┘
```

Operands

IP_address=*ip_address*
: Specifies the IP address, which can be configured on the system or it can be

specified with asterisks (*) as wild cards. The valid wildcard 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=*interface_name*
Specifies 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=*mask*
Specifies the subnet mask for the associated interface's IP address.

Destination_Addr=*address*
Specifies the IP address of the host at the remote end of this interface. This parameter is only valid for point-to-point links. If this parameter is not specified for a point-to-point link, a route to the host at the remote end of the interface will not be added to the TCP/IP route table. A subnet route for the interface will be added at MPROUTE initialization independent of whether this parameter is specified.

MTU=*size*
Specifies the mtu size for MPROUTE to add to the routing table for routes that take this interface. Valid values are 0 to 65535.

Max_Xmit_Time=*max_time*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 999.990.

Min_Xmit_Time=*min_time*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 99.990.

RT_Gain=*rtgain*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 1.0.

Variance_Gain=*vgain*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 1.0.

Variance_Mult=*mult*
Specifies 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 acknowledgment before resending a packet. Valid values are 0 to 99.990.

Delay_Acks
: Specifies the delay acknowledgments value to add to the routing table for routes that take this interface. Specifying YES delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specifying NO results in acknowledgments being returned immediately. Valid values are YES and NO.

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: Native ATM networks allow IP to use the network as a Non-Broadcast Multi- 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.
- *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, or channel-to channel 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 not 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 *priority router* 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 the higher router priority 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

OSPF Configuration Steps

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 "OSPF Router IDs".)
2. Define OSPF areas attached to the router. If no OSPF areas are defined, a single backbone area is assumed. (See "Define 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 "Define OSPF Interfaces" on page 214.)
4. If the router interfaces to non-broadcast networks (X.25, Frame-Relay or ATM Native), set additional interface parameters. (See "Set Non-Broadcast Network Interface Parameters" on page 216 and "Configuring Wide Area Subnetworks" on page 216.)
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 Autonomous System Boundary Routing" on page 218.)

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.

Define Backbone and Attached OSPF Areas

Figure 15 on page 212 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.

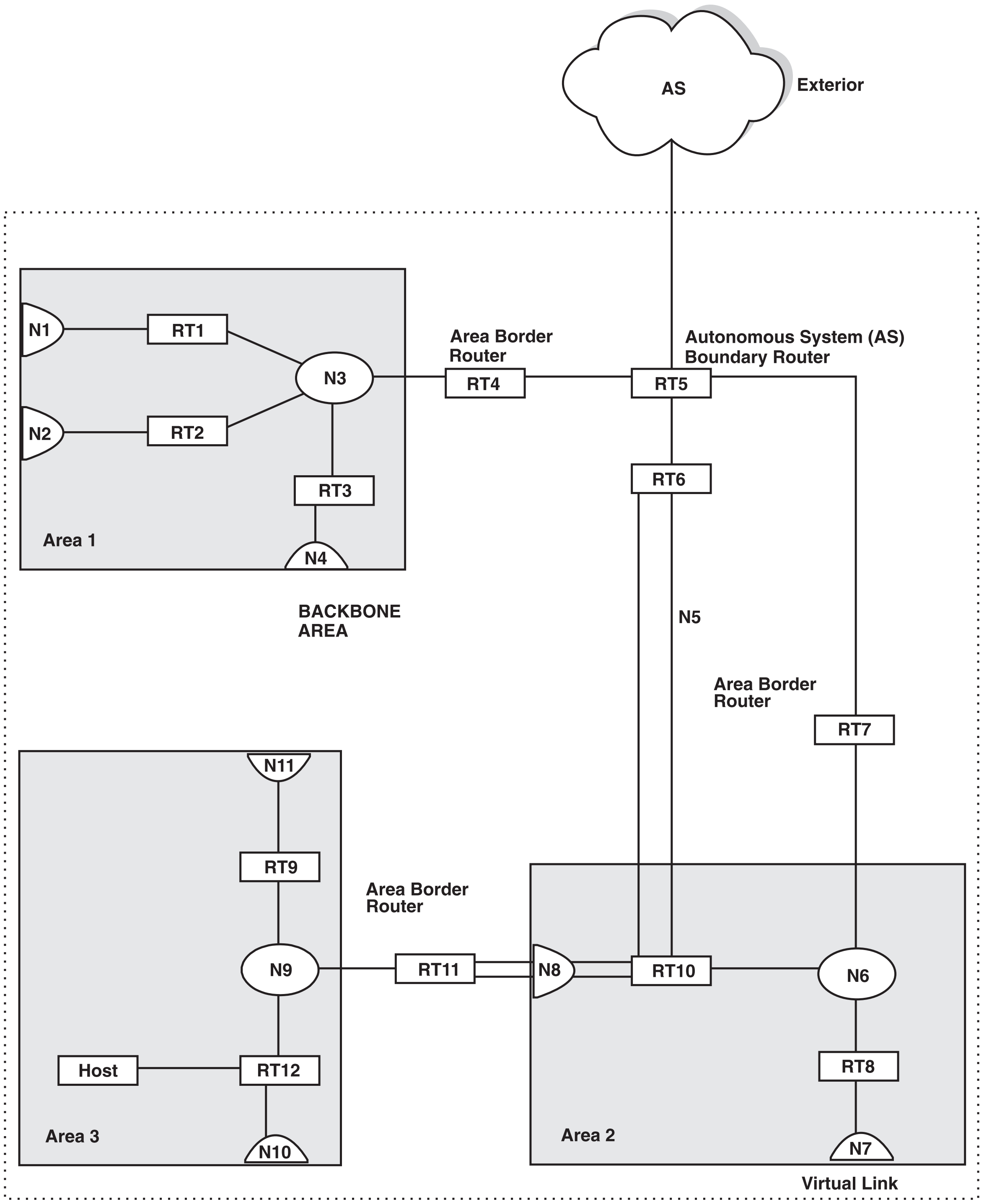


Figure 15. OSPF Areas

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 originally from ASE boundary routers, and propagated 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 may define an area as a stub when:
1. There is no requirement for the area to handle transit backbone traffic.
2. It is acceptable for area routers to use an area-border-router-generated default for traffic destined outside the AS.
3. 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.

Define 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 216.

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.

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 155 Mbps ATM.

Table 14. 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 an SMSG command. This new cost is propagated 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.

Notes:

1. 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).
2. OSPF routers within the same IP subnetwork must be configured as boundary routers (using the AS_BOUNDARY_ROUTING statement for MPROUTE and using a corresponding configuration option on the adjacent OSPF router) or the routers will fail to form direct connections and therefore fail to exchange routing tables.

Set 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 Autonomous System 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. See "Configuring for Routing Protocol Comparisons" on page 219 for an explanation of metric types.

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 Native ATM, the OSPF configuration options are the same as for Wide Area Subnetworks. See "Configuring Wide Area Subnetworks" on page 216. Both NBMA and Point-to-Multipoint configurations are supported.

Configuring OSPF Over Native ATM

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

Logical IP Subnet needs to be DR; however, if other routers are also configured to be DR-eligible, the Logical IP Subnet 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 16). 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.

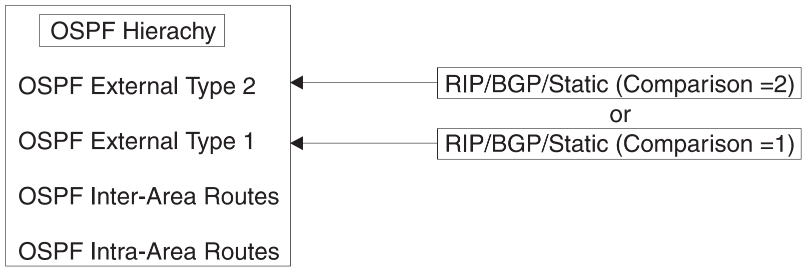


Figure 16. OSPF Routing Hierarchy

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

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 9. Configuring the DNS Server

The Domain Name System (DNS) server — commonly referred to as simply a *name server* — maps a host name to an internet address or an internet address to a host name. To configure the DNS server virtual machine, you must perform the following steps:

DNS Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the DNS server.
3. Determine the type of DNS server to configure. The section titled "DNS Server Overview" may help in this determination.
   - Configure a Caching-Only name server:
     a. Customize the DNS server configuration file to include a **CACHINGONLY** statement.
     b. Identify one or more remote name servers using cache file resource records.
   - Configure a Primary name server:
     a. Customize the DNS server configuration file to include **PRIMARY** and **SECONDARY** statements.
     b. Prepare the DB2 Server for VSE & VM database.
     c. Define the DB2 database.
     d. Define the appropriate resource records for your installation.
     e. Create a MASTER DATA file.
     f. Install the name server database.

Dynamic Server Operation

The DNS server provides a VM Special Message (SMSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information see "Dynamic Server Operation" on page 245.

DNS Server Overview

A Domain Name System (DNS) server maps a host name to an internet address or an internet address to a host name. The domain name server (or more simply, the *name server*) is like a telephone book that contains a person's name, address, and telephone number. For each host, the name server can maintain internet addresses, nicknames, mailing information, and available well-known services (for example, DNS, SMTP, FTP, or Telnet). This information is maintained through the use of resource records.

When a client needs to communicate with another host, the client uses either the internet address of the remote host or sends a query to the Domain Name System (DNS) for host name resolution.

However, before the name server can resolve a query, it must be supplied with resource records that either define a zone, or identify a different (remote) name server that can provide the information required for a response. For information about resource record format, see "Resource Records" on page 240.

Primary Versus Caching-Only Name Servers

If a name server maintains local data and has authority for the zone defined by this data, it is called a *primary* name server. If a name server zone transfers a zone from a remote primary name server, it is called a *secondary* name server. Once a secondary name server receives zone data, it has authority for that zone. The secondary name server then periodically refreshes this zone data based on values defined in the zone's authority record.

For more information about primary and secondary zones, see RFC 1034 and 1035.

A name server that does not have authority for any zone is called a *caching-only* name server. To respond to queries, a caching-only name server must communicate with a remote name server in the internet that has access to zone data.

The TCP/IP domain name server can be configured to run as a primary, secondary, or caching-only server. Both primary and secondary name servers store zone data in DB2 tables. To set up a name server that does *not* use DB2, configure the domain name server as a caching-only name server. The caching-only name server uses a CMS file to define the host names and internet addresses for the remote name servers. For more information, see "CACHINGONLY Statement" on page 228.

The manner in which the name server operates is controlled by statements defined in a DNS configuration file. This file is described in more detail in "DNS Configuration File Statements" on page 228.

Host Name Resolution

TCP/IP applications have a resolver routine compiled into their code that resolves host names into internet addresses. This code accesses a name server, site tables, or both, depending on your configuration. You can change the configuration in the TCPIP DATA file. For information about the TCPIP DATA file, see "Chapter 5. Defining the TCP/IP System Parameters" on page 139.

The VM resolver accesses the site tables only in the following cases:

- You have not defined a name server in the TCPIP DATA file.
- The name server either does not respond, or the name server responds with an `unknown host` error. The only exception is SMTP, which never uses the site tables if it is configured to use a name server.

The VM name server has three main locations for retrieving data: cache memory, a DB2 database, or other name servers.

The name server caches query answers in one of its four caches. All future queries are answered directly from the cache without referencing the DB2 database or another name server. Records remain active in the cache, based on the time-to-live (TTL) field for each of the records. If a cache becomes full, the least recently used entry is deleted, if it has been in the cache for the number of seconds specified using the LRUTIME statement. For more information, see "LRUTIME Statement" on page 231.

If the name server has authority for the query's zone, the name server consults its DB2 database for the query answer. This answer is then sent back to the resolver and cached for future queries.

The name server communicates with other name servers to query records outside its zone, to answer queries about zones for which it has authority, and to transfer zones both to and from other name servers. To contact other name servers, the addresses of these name servers must be made available. This information is held in a root cache file. The name of this file is supplied on the CACHINGONLY statement or the ROOTCACHE statement in the name server configuration file. A sample root cache file is supplied and is named NSMAIN SCACHE.

If this name server is behind a firewall, it may not be able to directly contact the other name servers it would like to in order to process its questions. In that case, a FORWARDERS statement can be used to define to this name server the addresses of any firewall name servers or other name servers that can access name servers outside the firewall.

Figure 17 on page 226 illustrates the domain name resolution process.

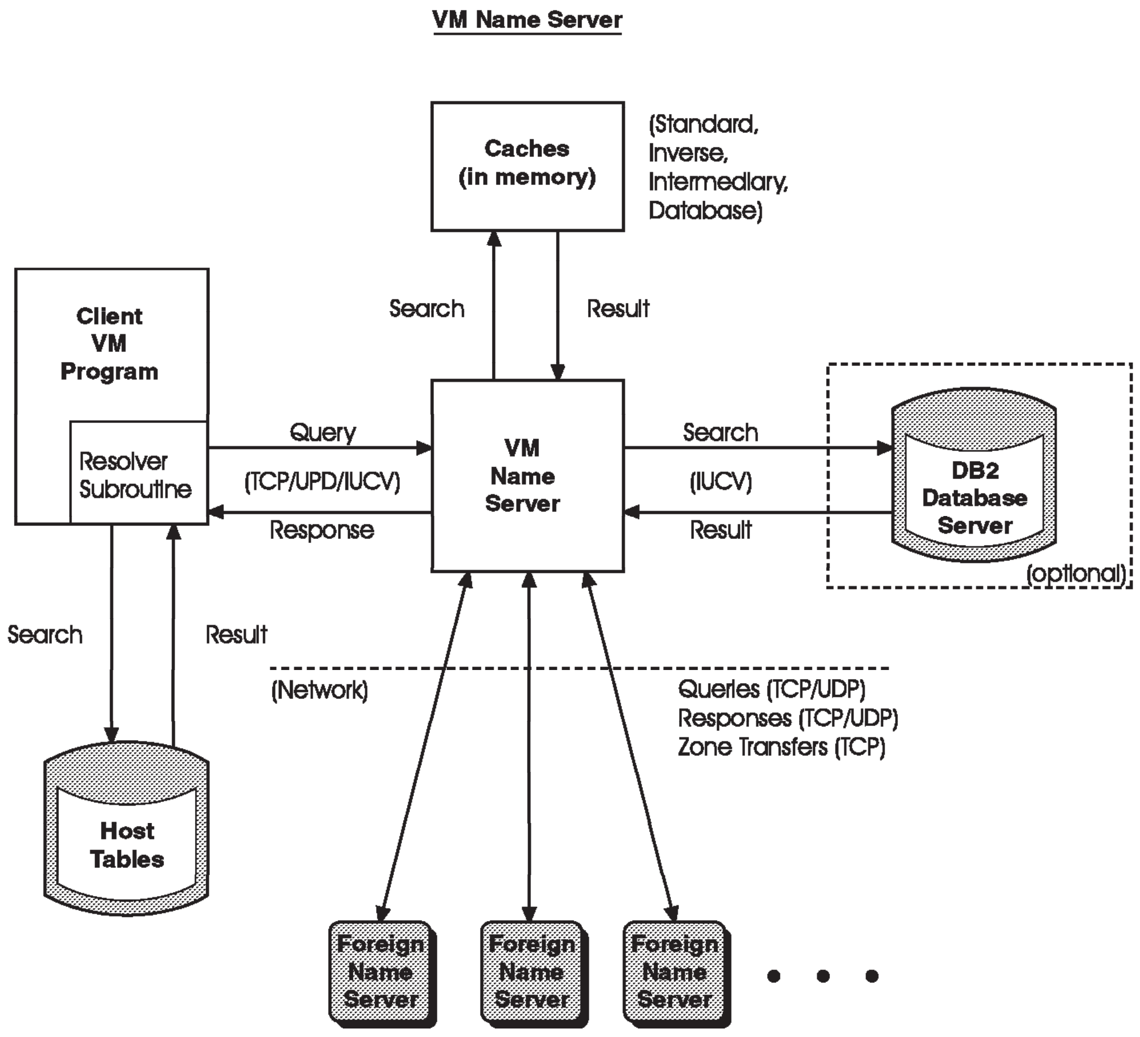


Figure 17. VM Domain Name Resolution

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  NAMESRV  0
```

The name server requires ports TCP 53 and UDP 53 to be reserved for it. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
  53 TCP NAMESRV     ;   DNS Server
  53 UDP NAMESRV     ;   DNS Server
```

Step 2: Update the DTCPARMS File

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

```
:nick.NAMESRV
  :PARMS.
  :DB2_DATABASE.
```

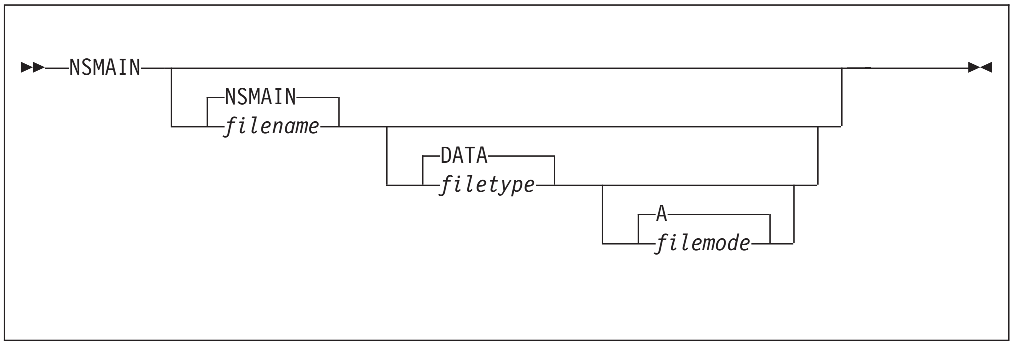
If more customizing is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the name server if you:
- Use a configuration file other than NSMAIN DATA.
- Configure a primary name server and need to identify a DB2 database to be referenced.

NSMAIN Command

DNS services are initiated using the NSMAIN command:



Specify NSMAIN command operands as **:Parms.** tag start-up parameters in your DTCPARMS file.

Operands

filename
: The file name of the DNS server configuration file. The default file name is NSMAIN.

filetype
: The file type of the configuration file. The default file type is DATA.

filemode
: The file mode of the configuration file. The default file mode is A.

Step 3: Customize the NSMAIN DATA File

The NSMAIN DATA configuration file, NSMAIN DATA, defines how the DNS server is to operate. This file allows you to specify:
- Internet addresses
- Nicknames for fully qualified domain names
- Mailing information

- DB2 tables
- Trace options

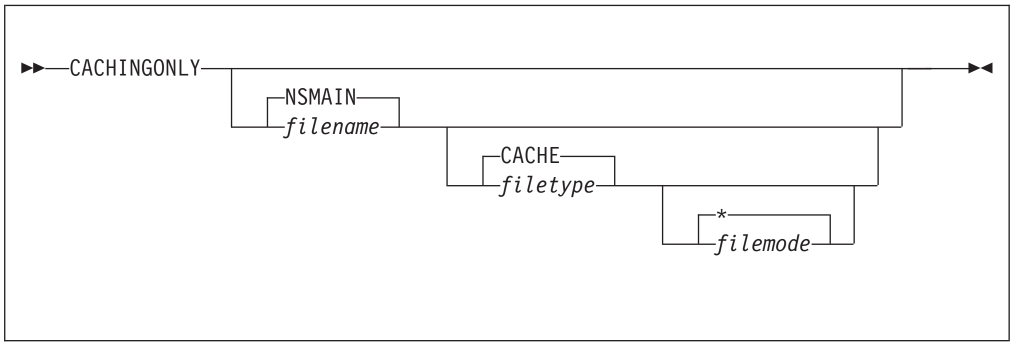
See “DNS Configuration File Statements” for detailed information about how to specify entries within this file. A sample DNS configuration file is provided as NSMAIN DATA on the TCPMAINT 591 disk. Your customized DNS configuration file should be copied to the TCPMAINT 198 minidisk and renamed to match what is specified in the NSMAIN command (the default name is NSMAIN DATA).

DNS Configuration File Statements

This section describes the statements used to configure the Domain Name Server.

CACHINGONLY Statement

The CACHINGONLY statement specifies the name server is to operate in caching-only mode. The file specified by this statement contains information (type NS and A resource records) necessary to communicate with remote name servers. These remote name servers are usually internet or intranet root name servers.



Operands

filename
: The file name of the cache file that contains the resource records to be used.

filetype
: The file type of the cache file.

filemode
: The file mode of the cache file.

Examples

To use a cache file named **MYCACHE DNSINFO**, specify the following in the DNS server configuration file:

```
CACHINGONLY MYCACHE DNSINFO
```

The content of the MYCACHE DNSINFO file might be:

```
EDU              608400  IN NS TERP.UMD.EDU
TERP.UMD.EDU     608400  IN A  128.8.10.90
.                608400  IN NS C.NYSER.NET
C.NYSER.NET      608400  IN A  192.33.4.12
```

In this example, queries for the EDU domain are sent to TERP.UMD.EDU (IP address 128.8.10.90). All other queries initially go to the root server, C.NYSER.NET (IP address 192.33.4.12).

Note: A sample cache file is provided on the TCPMAINT 591 disk as NSMAIN SCACHE. If needed, this file should be copied to the TCPMAINT 198 minidisk, customized, and renamed to match the file name and type specified by the CACHINGONLY statement in the DNS server configuration file (the provided NSMAIN SDATA sample specifies NSMAIN CACHE).

Usage Notes

1. You must create the cache file named by the CACHINGONLY statement, and add the information required for the caching-only name server to communicate with the root name server(s). The information in this file a combination of name server (type NS) resource records and their corresponding (type A) IP address resource records.
2. Record types other than NS and A, such as MX or CNAME, should not be specified in the cache file. If such records are present, they will be ignored.
3. If your data resides in a DB2 table, this statement should not be used. Instead, use the ROOTCACHE statement to provide root name server data when the name server is operating as a PRIMARY or SECONDARY name server.
4. If you have either PRIMARY or SECONDARY statements in the configuration file, as well as a CACHINGONLY statement, the name server will not start.
5. The time to live (TTL) field of resource records specified in the cache file is ignored when these records are processed.
6. For more information about resource records, see the *TCP/IP User's Guide* or a suitable Domain Name System reference.
7. When the caching-only name server receives a query, it will look for a response in its cache. If found, that response is returned to the client requestor. If not found, the query will be forwarded, as needed, to a remote name server listed in the cache file.

   When a query response arrives at the caching-only name server, it is cached and returned to the client. This response will remain in the cache for as long as the time-to-live (TTL) field in the response dictates.

CHECKORIGIN Statement

Some resolvers incorrectly implement search lists (see note below) causing a duplicate domain origin to be appended to the query name. The CHECKORIGIN statement causes the name server to check the query name for a duplicate of the last label. This option was created for the instances when the Name Server is being bombarded with these queries. The statement is supplied as a single token in the name server configuration file that is processed by the NSMAIN command (default name: NSMAIN Data). The CHECKORIGIN statement has no parameters. No duplicate checking is performed if CHECKORIGIN does not appear in the file.

```
►►──CHECKORIGIN──────────────────────────────────────────────────────►◄
```

Note: A description of search lists can be found in RFC 1123, section 6 1.4.3.

DATABASEQUERYCACHE Statement

The DATABASEQUERYCACHE statement is no longer used. If it is present in the configuration file, it is ignored.

DOMAINNAMEPORT Statement

The DOMAINNAMEPORT statement changes the port number that the name server uses. This parameter should be changed only for debugging purposes.

```
   ┌─DOMAINNAMEPORT 53──────────┐
►►─┼────────────────────────────┼──────────────────────────────────►◄
   └─DOMAINNAMEPORT port_number─┘
```

Operands

port_number
: An integer in the range of 1 through 65 535 that specifies the port number that the name server uses for all TCP and UDP communications. The default port number is 53.

FORWARDERS Statement

Use the FORWARDERS statement when the name server is behind a firewall that prevents the name server from contacting other name servers. This statement directs the name server to send any question that it can't answer to the name server(s) at the IP address(es) specified. These name servers should be recursive.

```
                 ┌──────────────────────┐
                 ▼                      │
►►──FORWARDERS──────internet_address────┴──────────────────────────►◄
```

Operands

internet_address
: Specifies the internet address of a remote name server that should handle questions when this name server doesn't know the answer. This could be a nearby recursive name server or a firewall name server. If multiple name servers can be specified, there will be less chance of failure should one of the name servers not be available.

HOSTNAMECASE Statement

The HOSTNAMECASE statement defines the case to which the host names in all queries are translated in order to match the contents of the name server DB2 database. The contents of the DB2 Server for VSE & VM database are assumed to be in the case specified by this statement. If they are not, the name server will not operate correctly.

Note: This statement pertains only to DB2 database queries.

```
►►──HOSTNAMECASE──┬─UPPER─┬──────────────────────────────────────────────►◄
                  └─LOWER─┘
```

Operands

UPPER
Specifies that the data in the DB2 database is in UPPER case.

LOWER
Specifies that the data in the DB2 database is in LOWER case.

INTERMEDIARYQUERYCACHE Statement

The INTERMEDIARYQUERYCACHE statement is no longer used. If it is present in the configuration file, it is ignored.

INVERSEQUERYCACHE Statement

The INVERSEQUERYCACHE statement is no longer used. If it is present in the configuration file, it is ignored.

LRUTIME Statement

The LRUTIME statement specifies the amount of time a record must be in the cache before it can be replaced by a new cache entry.

The name server uses a least recently used (LRU) algorithm to replace entries in a full cache.

```
   ┌─LRUTIME 300─────┐
►►─┼─────────────────┼──────────────────────────────────────────────────►◄
   └─LRUTIME seconds─┘
```

Operands

seconds
The number of seconds after which cache entries should be replaced. The default time is 300 seconds (5 minutes).

MSGNOH Statement

The MSGNOH statement specifies that the name server use the CP MSGNOH command to reply to an SMSG command. Otherwise, the name server uses the CP MSG command.

Note: The CP MSGNOH command is a privileged command. Ensure that your name server is authorized to issue this command if you are using the MSGNOH statement.

```
►►──MSGNOH──────────────────────────────────────────────────────────►◄
```

The MSGNOH statement has no operands.

NEGATIVECACHING Statement

The NEGATIVECACHING statement specifies that the VM name server caches negative queries.

Negative caching prevents the name server from repeatedly searching for nonexistent resource records. The VM name server caches queries that do not contain answers, but have an SOA resource record in the additional section. The query response remains in the cache for the time specified in the minimum time-to-live (TTL) field of the SOA record. Subsequent queries for the same entry return the negative query response from the cache.

```
►►──NEGATIVECACHING─────────────────────────────────────────────────►◄
```

Operands

The NEGATIVECACHING statement has no operands.

NORECURSION Statement

The NORECURSION statement forces the name server to respond with either the query answer (if available) or with the remote name server host name and the corresponding internet address.

Normally, the name server does recursion by querying remote name servers for information outside the local domain.

```
►►──NORECURSION─────────────────────────────────────────────────────►◄
```

Operands

The NORECURSION statement has no operands.

PRIMARY Statement

The PRIMARY statement identifies a local zone for which the name server is responsible, as well as the name of a DB2 table where data about this zone is maintained.

```
►►──PRIMARY──zone_name──SQLbasetable────────────────────────────────►◄
```

Operands

zone_name
The name of the local zone.

SQLbasetable
The name of the DB2 base table where zone data is maintained.

For example, to define the authoritative table for the local `watson.ibm.com` zone, the following line is added to the DNS server configuration file:

```
PRIMARY watson.ibm.com  watson
```

Before starting a name server that uses the PRIMARY statement, you must execute the NSTABLE MODULE, with the file name and file type of the DNS server configuration file specified as parameters.

The NSTABLE MODULE creates two tables in the name server's dbspace using the *SQLbasetable* specified in the PRIMARY statement. The name server switches between the two tables. Queries are answered from one table until an SMSG FLIPTABLE command is received. For more information, see "SMSG FLIPTABLE Command" on page 246. At this point, the table name in the remark statement of the system.syscatalog is updated. The name server must be able to process queries from the old zone until the new zone is completely updated.

Also, before starting a name server that uses the PRIMARY statement, you must load the tables with data. The recommended way to do this is to use the NSDBLOAD MODULE.

ROOTCACHE Statement

The ROOTCACHE statement specifies the name of the file that contains information (type NS and A resource records) necessary to communicate with remote name servers. These remote name servers are usually internet or intranet root name servers.

```
►►──ROOTCACHE──┬──────────────────────────────────────────────────────┬──►◄
               │ ┌─NSMAIN───┐                                         │
               └─┴─filename─┴──┬───────────────────────────────────────┤
                               │ ┌─CACHE────┐                          │
                               └─┴─filetype─┴──┬──────────────────────┤
                                               │ ┌─*────────┐         │
                                               └─┴─filemode─┴─────────┘
```

Operands

filename
Specifies the file name of the cache file that contains the resource records to be used.

filetype
Specifies the file type of the cache file.

filemode
Specifies the file mode of the cache file.

Examples: To use a cache file named **MYCACHE DNSINFO**, specify the following in the DNS server configuration file:

```
ROOTCACHE MYCACHE DNSINFO
```

The content of the MYCACHE DNSINFO file might be:

```
EDU             608400  IN NS TERP.UMD.EDU
TERP.UMD.EDU    608400  IN A  128.8.10.90
.               608400  IN NS C.NYSER.NET
C.NYSER.NET     608400  IN A  192.33.4.12
```

In this example, queries for the EDU domain are sent to TERP.UMD.EDU (IP address 128.8.10.90). All other queries initially go to the root server, C.NYSER.NET (IP address 192.33.4.12).

Note: A sample cache file is provided on the TCPMAINT 591 disk as NSMAIN SCACHE. If needed, this file should be copied to the TCPMAINT 198 minidisk, customized, and renamed to match the file name and type specified by the ROOTCACHE statement in the DNS server configuration file (the NSMAIN SDATA sample provided specifies NSMAIN CACHE).

Usage Notes

1. You must create the cache file named by the ROOTCACHE statement, and add the information required for the only name server to communicate with the root name server(s). The information in this file a combination of name server (type NS) resource records and their corresponding IP (type A) address resource records.
2. Record types other than NS and A, such as MX or CNAME, should not be specified in the cache file. If such records are present, they will be ignored.
3. The time to live (TTL) field of resource records specified in the cache file is ignored when these records are processed.
4. For more information about resource records, see the *TCP/IP User's Guide* or a suitable Domain Name System reference.
5. When the name server receives a query, it will look for a response in its cache. If found, that response is returned to the client requestor. If not found, the query will be forwarded, as needed, to a remote name server. If the name server has no better idea who to ask for an answer, it will use a name server listed in the cache file.

   When a query response is returned to the name server, it is cached and returned to the client. This response will remain in the cache for as long as the time-to-live (TTL) field in the response dictates.

SECONDARY Statement

The SECONDARY statement identifies a zone for which information will be transferred from a remote location, as well as the name of a DB2 table where this zone data is maintained.

```
►►──SECONDARY──zone_name──SQLbasetable──internet_address─────────────────►◄
```

Operands

zone_name
The name of the zone to transfer.

SQLbasetable
The name of the DB2 base table where transferred zone data is maintained.

internet_address
The internet address of the remote name server from which zone data is transferred.

For example, to zone transfer the `raleigh.ibm.com` and `phoenix.ibm.com` zones, the following lines are added to the DNS server configuration file:

```
SECONDARY raleigh.ibm.com   raleigh  9.67.43.126
SECONDARY phoenix.ibm.com   phoenix  9.4.1.2
```

Before starting a name server that uses the SECONDARY statement, you must execute NSTABLE MODULE, with the file name and file type of the DNS server configuration file as parameters.

The NSTABLE MODULE creates two tables in the name server's dbspace using the *SQLbasetable* specified in the SECONDARY statement. The name server switches between the two tables. Queries are answered from one table until a new zone is completely received in the other table. At this point, the table name in the remark statement of the system.syscatalog is updated. The name server must be able to process queries from the old zone until the new zone is completely received.

The table specified in a SECONDARY statement is loaded by the name server, via the network, from the name server at the address specified by the *internet_address*.

SMSGUSERFILE Statement

The SMSGUSERFILE statement specifies the name of an exec used to verify whether a user ID is authorized for a particular name server SMSG command. The authorization exec is passed the following arguments, in order:

- the node ID of the system from which the SMSG command originated
- the user ID that issued the SMSG command
- the SMSG command

Although the authorization exec has a node parameter, there is currently no support for SMSG command processing from remote nodes. The parameter exists to maintain interface compatibility across releases.

The authorization exec, if used, must return a value of **0** if the user is allowed to perform the SMSG command, and should return a **1** if the user is not to be allowed to perform the command. The HELP, LIST, and STATS commands are commonly accepted as the only SMSG commands that are not privileged. For information about SMSG commands, see "SMSG Interface to the DNS Server" on page 245.

```
                    ┌─VALIDUSR─┐
►►──SMSGUSERFILE──┴─filename─┴──────────────────────────────────────►◄
```

Operands

filename
The file name of the verification exec. The first exec of this name found in the CMS search order is used. The default file name is VALIDUSR.

A sample verification file is shipped as VALIDUSR SEXEC on the TCPMAINT 591 disk. If needed, this file should be copied to the TCPMAINT 198 disk, customized, and renamed to match what is specified in the DNS server configuration file. (The provided NSMAIN SDATA sample specifies VALIDUSR).

Usage Notes

1. If the SMSGUSERFILE statement is omitted, SMSG authorization will still be attempted, using the VALIDUSR default.
2. The authorization exec can be changed while the name server is running. The name server refreshes its access to all file modes before it calls the authorization exec. It will have access to any updates made to the file.
3. If no SMSG authorization exec exists, all SMSG command requests will be rejected. In this case the following message will be displayed on the name server console for each SMSG request:

   ```
   DMSEXT072E Error in EXEC file VALIDUSR, line 0 - not found
   ```

STANDARDQUERYCACHE Statement

The STANDARDQUERYCACHE statement allows the specification of the number of standard queries and answers to be managed by the cache.

The query and the resulting answer are stored in a cache. Query answers that are cached do not access the DB2 database and are not sent to remote name servers. This results in faster responses to the client.

```
      ┌─STANDARDQUERYCACHE 3000────┐
►►────┼────────────────────────────┼──────────────────────────────────►◄
      └─STANDARDQUERYCACHE queries─┘
```

Operands

queries
: An integer value that represents the number of standard queries and answers to be managed by the cache. The default is 3000. A cache size of 1000 to 5000 entries is recommended to improve performance. The cache size is dynamically reduced if the server runs low on memory.

TRACE Statement

The TRACE statement establishes the set of trace categories for internal name server tracing. Name server trace output is displayed on the name server console. Most of the trace categories are for special diagnostic situations. The **[NO]** form of the trace category turns that specific trace category off. Thus, specifying TRACE ALL NOSUB NOSTOR NOMORE would enable all trace categories except storage and subroutine tracing, and would disable the trace points that require the MORE trace category.

```
            ┌─────────────────────┐
            ▼                     │
►►──TRACE───┬─ALL─────────────┬───┴──────────────────────────────►◄
            ├─END─────────────┤
            ├─[NO]AUth────────┤
            ├─[NO]CAche───────┤
            ├─[NO]DBG─────────┤
            ├─[NO]DBG2────────┤
            ├─[NO]INit────────┤
            ├─[NO]IUcv────────┤
            ├─[NO]MOre────────┤
            ├─[NO]NOtice──────┤
            ├─[NO]PArms───────┤
            ├─[NO]PKTIn───────┤
            ├─[NO]PKTOut──────┤
            ├─[NO]QUeue───────┤
            ├─[NO]SQl─────────┤
            ├─[NO]STorage─────┤
            ├─[NO]SUbroutines─┤
            ├─[NO]TCp─────────┤
            ├─[NO]TYpe────────┤
            ├─[NO]UDp─────────┤
            ├─[NO]UTl─────────┤
            └─[NO]ZOne────────┘
```

Operands

ALL
: All trace categories are enabled. This results in extensive console tracing.

END
: All trace categories are disabled.

AUth

NOAUth
: Enables or disables console tracing of authority determination.

CAche

NOCAche
: Enables or disables console tracing of caching activities.

DBG

NODBG
: Enables or disables console tracing in special diagnostic circumstances.

DBG2

NODBG2
: Enables or disables console tracing in special diagnostic circumstances.

INit

NOINit
: Enables or disables console tracing of initialization activities.

IUcv

NOIUcv
: Enables or disables console tracing of IUCV activity.

| **MOre**

| **NOMOre**
| Enables or disables additional trace points that are normally skipped because
| of the large amount of trace data displayed.

| **NOtice**

| **NONOtice**
| Enables or disables console tracing of TCP/IP message notifications.

| **PArms**

| **NOPArms**
| Enables or disables console tracing of input parameters to most subroutines.

| **PKTIn**

| **NOPKTIn**
| Enables or disables console tracing of most inbound DNS messages received.

| **PKTOut**

| **NOPKTOut**
| Enables or disables console tracing of most inbound DNS messages sent to
| other hosts.

| **QUeue**

| **NOQUeue**
| Enables or disables console tracing of questions asked of this name server and
| questions and answers related to those questions.

| **SQl**

| **NOSQl**
| Enables or disables console tracing of various activities involving the DB2
| database.

| **STorage**

| **NOSTorage**
| Enables or disables console tracing of storage usage.

| **SUbroutines**

| **NOSUbroutines**
| Enables or disables console tracing of each subroutine executed. Some utility
| subroutines are instead traced with the UTl trace.

| **STorage**

| **NOSTorage**
| Enables or disables console tracing of storage usage.

| **TCp**

| **NOTCp**
| Enables or disables console tracing of activities using the TCP protocol.

| **UDp**

| **NOUDp**
| Enables or disables console tracing of activities using the UDP protocol.

| **UTl**

NOUTl
: Enables or disables console tracing of input parameters to and results from many utility subroutines.

ZOne

NOZOne
: Enables or disables console tracing of zone transfer requests, and zone transferring of local zones.

UDPONLY Statement

The UDPONLY statement instructs the name server to perform recursive name resolution to remote name servers using only the UDP protocol. Without this statement, the name server attempts to communicate with remote name servers using UDP, and if that fails, it tries again using the TCP protocol. The use of UDPONLY is not recommended.

```
►►──UDPONLY──────────────────────────────────────────────────────────►◄
```

Operands

The UDPONLY statement has no operands.

UDPRETRYINTERVAL Statement

The UDPRETRYINTERVAL statement instructs the name server to wait a specified amount of time before attempting to communicate with another authoritative name server.

The name server attempts to communicate with remote name servers using UDP. If a remote name server fails to respond, the name server attempts to communicate with another remote name server.

```
      ┌─UDPRETRYINTERVAL 5───────┐
►►────┼──────────────────────────┼───────────────────────────────────►◄
      └─UDPRETRYINTERVAL seconds─┘
```

Operands

seconds
: The number of seconds to wait before sending the query to the next remote name server. The default is 5 seconds.

Prepare the DB2 Database

The NSPREP EXEC has been included to prepare your DB2 database for the new name server. This EXEC allows access to the DB2 database by the name server programs, which have been written using the SQL/DS Version 3 Release 5, or DB2 Server for VSE & VM Version 5 Release 1 C language interface.

```
►►──NSPREP──────────────────────────────────────────────────────────►◄
```

NSPREP preprocesses the following source files (with file types CSQL or ASMSQL) into your DB2 database.

File	**Description**
NSACQ	Acquires a database space (dbspace). The default number of pages is 5120.
NSTABLE	Creates two tables for each primary and secondary domain defined in the initialization file.
NSDBLOAD	Creates resource records defining a domain from a master file. The resource records are then inserted into a DB2 table.
NSDBSQL	Contains a subroutine of the NSMAIN MODULE. It uses the assembler DB2 interface. (This file type is ASMSQL.)

NSPREP also pre-processes the NSAXSUB1, NSINSERT, and NSSQL files.

Invoke NSPREP within the DNS server virtual machine.

Define the DB2 Database

The name server's DB2 database should be defined with the help of the DB2 database administrator. The DB2 Server for VSE & VM database administrator must acquire a dbspace for the name server by executing the NSACQ MODULE.

Note: DB2 Version 3 Release 5, or DB2 Server for VSE & VM Version 5 Release 1 or later, is required if you are using a DB2 database.

The types of SQL commands that the name server generates for each query type are:

Query Type	**Command**
Standard	SELECT * FROM *sqltable* WHERE TYPE=? AND CLASS=? - AND NAME=?
Inverse	SELECT NAME, TTL FROM *sqltable* WHERE TYPE=? AND - CLASS=? AND RDATA=?
Database	SELECT RDATA FROM *sqltable* WHERE TYPE=? AND - NAME=?
Database Update	UPDATE *sqltable* SET RDATA=? WHERE TYPE=? AND - NAME=?

Note: Completion queries are no longer supported.

Resource Records

After setting up the database as described in "Define the DB2 Database", you must load the data into the SQL table. These data entries are called resource records. NSDBLOAD MODULE can assist you in inserting the data into the table. The following is the format of a resource record that is used as input for the NSDBLOAD MODULE.

```
►►──name──┬─────┬──┬───────┬──recordtype──recorddata──────────────────►◄
          └─ttl─┘  └─class─┘
```

Operands

name
: The location in the dbspace for the resource record, such as the owner. If a resource record line starts with a blank, it is loaded into the location specified by the most recent location specifier. The location specifier is relative to an origin provided on the $ORIGIN record.

ttl
: The time-to-live (TTL) field, which is optional. This field is expressed as a decimal number. If it is omitted, the *ttl* default value is specified in the Start of Authority (SOA) record. TTL specifies, in seconds, how long the data remains in the cache.

class
: The class of the data.

recordtype
: The type of the data. This field describes the kind of data that exists in the next field, *recorddata*.

The following record types are recognized:

A	Address
NS	Name Server
CNAME	Canonical Name (nickname, alias)
SOA	Start of Authority
WKS	Well Known Services
PRT	Domain Name Pointer
HINFO	Host Information
MINFO	Mailbox Information
MX	Mail Exchanger
TYPE97	Database Update Authority
TXT	Text String

recorddata
: Depends on the values of the resource record class and type. The fields that make up the record data, are usually expressed as decimal numbers or as domain names.

Create a MASTER DATA File

MASTER DATA files are text files that contain resource records in text form. For more information about resource records, see “Resource Records” on page 240. MASTER DATA files, stored on TCPMAINT 198, are used to define a zone because the contents of a zone can be expressed as a list of resource records. MASTER DATA files can also be used to list the contents of a cache.

The format of these files is a sequence of entries. Entries are line-oriented, but you can use parentheses to continue a list across a line boundary, and text literals can contain Carriage Return Line Feeds (CRLF) within the text. You can use any combination of tabs and spaces as a delimiter between the items that make up an entry. You can end any line with a comment. The comment starts with a semicolon (;). You can use blank lines, with or without comments, anywhere in the file.

Two control entries are defined:

Entry	Definition
$ORIGIN	Followed by a domain name, and resets the current origin for relative domain names to the stated name.
$INCLUDE	Inserts the named file into the current file, and can specify a domain name that sets the relative domain name origin for the included file. The use of $INCLUDE is optional.

For a complete description of master files, see RFC 1034 and RFC 1035.

Some characters have special meanings:

Character	Description
@	A free-standing @ denotes the current origin.
•	One free-standing dot represents the null domain name of the root.
\X	*X* is any character other than a digit. Use \X to quote this character so that its special meaning does not apply, as in a mail box specification, or an SOA record. For example, you can use a backslash followed by a dot (`\.`) to place a dot character in a label.
( )	Parentheses group the data that crosses a line. Line terminations are not recognized within parentheses.
;	A semicolon begins a comment. The remainder of the line is ignored.

Each zone must contain an SOA and NS resource record. The following is an example of a master domain file:

```
;The following $ORIGIN record will be (IBM.COM.) appended to any
;name that does not end in a dot until the next $ORIGIN record.
;*********************************************************************
;             Authoritative for Domain IBM.COM
;*********************************************************************
$origin IBM.COM.         ; THE ORIGIN FOR ALL NAMES
ibm.com.  IN  SOA  yktvmx.watson.ibm.com.  zohar.yktvmx.watson.ibm.com.  (
                        091690    ;serial number for data
                        21600     ;refresh value for secondary NS (in secs)
                        3600      ;retry value for secondary NS (in secs)
                        360000    ;expire data when refresh not available (secs)
                        86400 )   ;minimum time to live value (secs)
ibm.com.  IN   NS   yktvmx.watson.ibm.com.
ibm.com.  IN   NS   aides.watson.ibm.com.
ibm.com.  IN   NS   vmn.almaden.ibm.com.
$include ibm data a               ;File contain hosts addresses
```

The following is an example of an IBM data file:

```
raleigh.ibm.com.          IN    NS  ralvmm.raleigh.ibm.com.
ralvmm.raleigh.ibm.com.   IN    A   9.67.43.126
;
watson.ibm.com.           IN    NS  yktvmx.watson.ibm.com.
yktvmx.watson.ibm.com.    IN    A   129.34.128.246
```

The DNS defines a special domain called in-addr.arpa to translate internet addresses to domain names. An in-addr.arpa name is composed of the reverse octet order of an IP address concatenated with the in-addr.arpa string.

Install the Name Server Database

At this point, your DB2 database virtual machine has been created. To acquire a dbspace in the database, log on to the name server virtual machine, and execute the NSACQ MODULE. The NSACQ MODULE issues the SQL ACQUIRE DBSPACE command. The name server must have the authority granted to execute the NSACQ command. The authority can be acquired by granting the name server Database Administrator (DBA) authority.

```
          ┌─TCPDBA─┐  ┌─TCPSPACE─┐  ┌─5120───┐
►►──NSACQ──┴─dbname─┴──┴─dbspace──┴──┴─npages─┴──────────────────────────►◄
```

Operands

dbname
: The name of the DB2 database set up for the name server by the database administrator. The default is TCPDBA.

dbspace
: The name of the database space set up for the name server. The default is TCPSPACE.

npages
: The size of the database machine. For example, you can choose 128, 256, 512, 1024, 2048, 5120, or 12 800. The default is 5120.

Insert Data in the Database

You must insert the resource records that define a zone into separate DB2 tables. To update a zone without shutting down the name server, you need two tables for each zone. Each of these tables contains the following fields:

```
NAME      varchar (150)
CLASS     smallint
TYPE      smallint
TTL       integer
RDATA     varchar (250)
```

For each primary zone, add a line to the name server configuration file containing the zone origin and base name (17 characters maximum) of the DB2 table. For each secondary zone, add a line to the name server configuration file containing the zone origin, the base name of the DB2 table to use, and the internet address of the remote name server. For example:

```
primary     watson.ibm.com.           watson
secondary   raleigh.ibm.com.          raleigh   9.67.43.100
```

After defining the primary and secondary zones in the configuration file, executing NSTABLE creates two DB2 tables by appending a `0` and `1` to the base name for each primary and secondary zone defined. The following tables are defined for the previous example:

```
watson0 watson1 raleigh0 raleigh1
```

```
►►──NSTABLE──┬──────────────────────────────────────────────────┬──────►◄
             │ ┌─NSMAIN───┐                                      │
             └─┴─filename─┴──┬──────────────────────────────────┬┘
                             │ ┌─DATA─────┐                     │
                             └─┴─filetype─┴──┬────────────────┬─┘
                                             │ ┌─A────────┐   │
                                             └─┴─filemode─┴───┘
```

Operands

filename
The file name of the DNS server configuration file that includes the PRIMARY and SECONDARY statements which identify the DB2 base table names that NSTABLE should construct. The default file name is NSMAIN.

filetype
The file type of the name server configuration file. The default file type is DATA.

filemode
The file mode of the name server configuration file. The default file mode is A.

When your database table is constructed, use the NSDBLOAD MODULE to load your database. NSDBLOAD reads the master data file you specify and creates a CMS file. The CMS file is read and the entries are loaded into the database.

After the name server is configured and running, you can use NSDBLOAD from another user to update the database. This user should have DB2 authority and be able to access the name server tables.

Once the tables are loaded from this user, an SMSG FLIPTABLE command can be issued to make the name server switch to the updated table. This eliminates the need to shutdown the name server for updates.

```
►►──NSDBLOAD──SQLtablename──────────────────────────────────────────────►

►──┬──────────────────────────────────────────────────────────────┬─────►
   │ ┌─MASTER─────────┐                                           │
   └─┴─input_filename─┴──┬──────────────────────────────────────┬─┘
                         │ ┌─DATA───────────┐                   │
                         └─┴─input_filetype─┴──┬──────────────┬─┘
                                               │ ┌─A──────────────┐ │
                                               └─┴─input_filemode─┴─┘

►──┬─────────────────────────────────────────────────────────────────┬──►◄
   │ ┌─NSMAIN──────────┐                                             │
   └─┴─output_filename─┴──┬────────────────────────────────────────┬─┘
                          │ ┌─HOSTINFO────────┐                    │
                          └─┴─output_filetype─┴──┬───────────────┬─┘
                                                 │ ┌─A───────────────┐ │
                                                 └─┴─output_filemode─┴─┘
```

Operands

SQLtablename
The name of the DB2 table. If you use the base name for the table,

NSDBLOAD will attempt to determine which table the name server is currently using and update the other table. If it cannot determine which table is in use, it will update the table that ends with a zero (`0`)— (for example, `watson0` or `raleigh0`). You can use the full table name (for example, `watson0`, `raleigh1`) and that table will be updated.

input_filename
: The file name of the master data file written in resource record format. The default file name is MASTER.

input_filetype
: The file type of the master data file. The default file type is DATA.

input_filemode
: The file mode of the master data file. The default file mode is A.

output_filename
: The file name of the master data file. The default file name is NSMAIN.

output_filetype
: The file type of the master data file. The default file type is HOSTINFO.

output_filemode
: The file mode of the master data file. The default file mode is A.

Dynamic Server Operation

| The VM Special Message Facility (SMSG) command provides an interactive
| interface to the DNS server to:
| • obtain information about the name server
| • diagnose name server problems using trace facilities
| • purge cache entries
| • change DB2 for VM tables
| • close the name server console log.

| See the "SMSGUSERFILE Statement" on page 235 for information on controlling
| authorization to issue SMSG commands to the name server.

SMSG Interface to the DNS Server

The various SMSG commands and operands supported by the name server are presented throughout the remainder of this section.

SMSG CLOSECON Command

The SMSG CLOSECON command causes the name server virtual machine to issue the command `CP SPOOL CONSOLE CLOSE`. The name server console is spooled to the owner identified in the DTCPARMS file.

```
►►──SMSG──server_id──CLosecon──────────────────────────────────────────►◄
```

Operands

server_id
: The user ID of the name server virtual machine.

SMSG COMMIT Command

The SMSG COMMIT command forces a commit of all DB2 transactions and releases the link to SQL.

```
►►──SMSG──server_id──COmmit──────────────────────────────────────────────────►◄
```

Operands

server_id
The user ID of the name server virtual machine.

SMSG DUMP Command

The SMSG DUMP command has been replaced by the SMSG STORAGE command. The SMSG DUMP command may be removed in a future release. It will accept the parameters of the SMSG STORAGE command and behave like the SMSG STORAGE command. This behavior is different than the behavior of the old SMSG DUMP command.

SMSG FLIPTABLE Command

The SMSG FLIPTABLE command informs the name server that new zone data appears in a DB2 table. Two DB2 tables are defined for each primary zone. NSDBLOAD can change the zone data in the DB2 table that the name server is not using. After the new data is in the DB2 table, you can issue an SMSG FLIPTABLE command to force the name server to use the updated DB2 table.

```
►►──SMSG──server_id──FLiptable──tablename────────────────────────────────────►◄
```

Operands

server_id
The user ID of the name server virtual machine.

tablename
The name of the table to flip. This is the base table name, and *not* the DB2 table name for example, (`watson`, not `watson0` or `watson1`).

SMSG HELP Command

The SMSG HELP command lists the SMSG commands supported by the name server.

```
►►──SMSG──server_id──HElp────────────────────────────────────────────────────►◄
```

Operands

server_id
The user ID of the name server virtual machine.

SMSG HINTS Command

The SMSG HINTS command allows you to display the addresses of the name servers that this name server will use when it does not know the answer to a question. The data comes from the FORWARDERS statement, if one is present in the configuration file. If a FORWARDERS statement is not present, the data comes from the contents of the file specified in the CACHINGONLY or ROOTCACHE statement.

```
                          ┌─Short─┐
►►──SMSG──server_id──HInts──┤       ├──────────────────────────────►◄
                          └─Long──┘
```

Operands

server_id
The user ID of the name server virtual machine.

Short
Indicates that only the IP addresses of the remote name servers are to be displayed.

Long
Indicates the IP addresses and the names of the remote name servers are to be displayed.

Usage Notes

1. If a FORWARDERS statement from the configuration file is in effect, all parameters on the SMSG HINTS command are ignored and only the IP addresses of the remote name servers will be displayed.

SMSG LEVEL Command

The SMSG LEVEL command allows you to display the service level of the name server and its component parts. The overall service level is provided in the response. The component part service information is directed to the name server console, and it can be obtained using the SMSG CLOSECON command. Output from a LISTFILE NSMAIN * * command is also included. This can be used to obtain the date and time of the NSMAIN MODULE in use and to ensure you are executing the intend level. If default configuration file names (NSMAIN DATA, NSMAIN CACHE) are in use, these will also be displayed by the LISTFILE command.

```
►►──SMSG──server_id──HElp──────────────────────────────────────►◄
```

Operands

server_id
The user ID of the name server virtual machine.

For example, if you enter `SMSG NAMESRV LEVEL`, the following information is displayed.

```
SMSG NAMESRV LEVEL

VM TCP/IP Name Server Level 320, service level PQ12345
```

SMSG LIST Command

The SMSG LIST command allows you to display the contents of the current name server caches.

```
                           ┌─STandard──┐
►►──SMSG──server_id──LIst──┼───────────┼──────────────────────────────►◄
                           ├─NScache───┤
                           ├─INverse───┤
                           ├─DBase─────┤
                           └─ALL───────┘
```

Operands

server_id
: The user ID of the name server virtual machine.

NScache
: This cache is no longer used. This operand is now ignored, but is accepted to maintain compatibility with prior levels of TCP/IP for VM.

STandard
: Lists the contents of the standard cache. This is the default. This operand is now ignored, but is accepted to maintain compatibility with prior levels of TCP/IP for VM.

INverse
: This cache is no longer used. This operand is now ignored, but is accepted to maintain compatibility with prior levels of TCP/IP for VM.

DBase
: This cache is no longer used. This operand is now ignored, but is accepted to maintain compatibility with prior levels of TCP/IP for VM.

ALL
: Lists the contents of all caches. Because the standard query cache is the only cache used, the parameter is equivalent to specifying STANDARD.

For example, if you enter `SMSG NAMESRV LIST ALL`, the following information is displayed:

```
----------------------------------------------------
Type Class Used Orig.TTL  Rem.TTL  Entry
Using standard query cache
A    IN       0   136347   130364  image.eimg.com
A    IN       0    86400    86235 -open
A    IN       0    86400    85244 -LOBO
A    IN       0    43200    42089 -ROGMO2@ENDICOTT.IBM.COM
PTR  IN       3    85802    57746  3.9.244.9.in-addr.arpa
PTR  IN       6    58298    56641  3.240.139.9.in-addr.arpa
A    IN       0    13056     6760  cnbcads.cnbc.com
A    IN       0    53561    43990  c.realtor.com
A    IN       6  9999999  9756054  dukhat.torolab.ibm.com
A    IN       1    21600    13120 -DS.INTERNIC.NET.ENDICOTT.IBM.COM
```

The following is a description of the fields that appear in the LIST command response:

Type
The resource record type.

Class
The resource record class.

Used
The number of times the record was returned from the cache.

Orig.TTL
The original value of the time-to-live field.

Rem.TTL
The remaining time that the record remains in the queue

Entry
The resource record name. Entries that are preceded by a negative sign (-) are cached negative queries. Negative entries indicate that the entity does not exist. If a query comes in for that entity, we will not forward the question to the name server for that domain, but will respond indicating that the entity does not exist. Negative entries are generated only when the NEGATIVECACHING statement has been specified in the configuration file. For more information, see “NEGATIVECACHING Statement” on page 232.

SMSG PURGE Command

The SMSG PURGE command purges one or all entries from all maintained caches.

```
►►──SMSG──server_id──PUrge──┬─entry_name─┬──────────────────────────►◄
                            └─ALL────────┘
```

Operands

server_id
The user ID of the name server virtual machine.

entry_name
A specific cache entry for which all occurrences should be purged from all caches.

ALL
Purges the contents of all caches.

SMSG REFRESH Command

The SMSG REFRESH command instructs the name server to immediately refresh the zone data contained in a DB2 table. This is used for a table for which the name server is acting as a secondary name server. It will cause the name server to attempt to contact the name server at the IP address specified on its associated SECONDARY configuration statement and reload the table with the zone data from that name server. It is useful when it is discovered that the primary name server has been updated and the data at our name server is obsolete.

```
►►──SMSG──server_id──REfresh──┬─zone_name─┬──────────────────────────────►◄
                              └─ALL───────┘
```

Operands

server_id
The user ID of the name server virtual machine.

zone_name
The name of the zone (`ibm.com`, to take an example from the sample configuration file) that should be refreshed. The *zone_name* must match the *zone_nam*e from a SECONDARY statement to have any effect. If the *zone_name* is not valid, no error message will be issued, but the command will not have any effect.

ALL
Causes the name server to refresh all zones for which it is acting as a secondary name server.

SMSG STATS Command

The SMSG STATS command returns useful information about the name server, such as start time, number of queries received, and size of cache.

```
►►──SMSG──server_id──STAts───────────────────────────────────────────────►◄
```

Operands

server_id
The user ID of the name server virtual machine.

For example, if you enter `SMSG NAMESRV STATS`, the following information is displayed.

```
SMSG NAMESRV STATS

NS start time:            Mon Oct  1 06:30:16
----------------------------------------------------
Total number of queries:  1632
Answers from cache:       435    (27%)
Size of cache:            150  used: 6
```

SMSG STORAGE Command

The SMSG STORAGE command dumps the internal storage management chain to the name server console. The data can then be recovered using the SMSG CLOSECON command.

```
►►──SMSG──server_id──STorage──Dump──────────────────────────────────────►◄
```

Operands

server_id
The user ID of the name server virtual machine.

Dump
Causes the internal storage management chain to be displayed on the name server console. If the Dump keyword is missing, no action is taken.

SMSG TRACE Command

The SMSG TRACE command starts or stops server tracing. The **[NO]** form of a trace category turns that specific trace category off.

Tracing is used primarily for problem diagnosis. The output from QUEUE tracing can also be useful in monitoring what hosts are using your name server and what other hosts they are attempting to access.

```
                                 ┌──────────────────┐
                                 ▼                  │
►►──SMSG──server_id──TRace──────┬─ALL──────────────┬┴─────────────────────►◄
                                ├─END──────────────┤
                                ├─[NO]AUth─────────┤
                                ├─[NO]CAche────────┤
                                ├─[NO]DBG──────────┤
                                ├─[NO]DBG2─────────┤
                                ├─[NO]INit─────────┤
                                ├─[NO]IUcv─────────┤
                                ├─[NO]MOre─────────┤
                                ├─[NO]NOtice───────┤
                                ├─[NO]PArms────────┤
                                ├─[NO]PKTIN────────┤
                                ├─[NO]PKTOUt───────┤
                                ├─[NO]QUeue────────┤
                                ├─[NO]SQl──────────┤
                                ├─[NO]STorage──────┤
                                ├─[NO]SUbroutines──┤
                                ├─[NO]TCp──────────┤
                                ├─[NO]TYpe─────────┤
                                ├─[NO]UDp──────────┤
                                ├─[NO]UTl──────────┤
                                └─[NO]ZOne─────────┘
```

Operands

server_id
The user ID of the name server virtual machine.

ALL
All trace categories are enabled. This results in extensive console tracing.

END
All trace categories are disabled.

AUth

NOAUth
Enables or disables console tracing of authority determination.

CAche

NOCAche
Enables or disables console tracing of caching activities.

DBG

NODBG
Enables or disables console tracing in special diagnostic circumstances.

DBG2

NODBG2
Enables or disables console tracing in special diagnostic circumstances.

INit

NOINit
Enables or disables console tracing of initialization activities.

IUcv

NOIUcv
Enables or disables console tracing of IUCV activity.

MOre

NOMOre
Enables or disables additional trace points that are normally skipped because of the large amount of trace data displayed.

NOtice

NONOtice
Enables or disables console tracing of TCP/IP message notifications.

PArms

NOPArms
Enables or disables console tracing of input parameters to most subroutines.

PKTIn

NOPKTIn
Enables or disables console tracing of most inbound DNS messages received.

PKTOut

NOPKTOut
Enables or disables console tracing of most inbound DNS messages sent to other hosts.

QUeue

NOQUeue
Enables or disables console tracing of questions asked of this name server and questions and answers related to those questions.

SQl

NOSQl
Enables or disables console tracing of various activities involving the DB2 database.

STorage

NOSTorage
Enables or disables console tracing of storage usage.

SUbroutines

NOSUbroutines
Enables or disables console tracing of each subroutine executed. Some utility subroutines are instead traced with the UTl trace.

STorage

NOSTorage
Enables or disables console tracing of storage usage.

TCp

NOTCp
Enables or disables console tracing of activities using the TCP protocol.

UDp

NOUDp
Enables or disables console tracing of activities using the UDP protocol.

UTl

NOUTl
: Enables or disables console tracing of input parameters to and results from many utility subroutines.

ZOne

NOZOne
: Enables or disables console tracing of zone transfer requests, and zone transferring of local zones.

SMSG VMDUMP Command

The SMSG VMDUMP command allows you to cause the name server to take a CMS DUMP if an abend occurs. It also allows you to take a DUMP immediately.

```
►►──SMSG──server_id──VMDump──┬─ON──┬──────────────────────────────────►◄
                             ├─OFF─┤
                             └─NOW─┘
```

Operands

server_id
: The user ID of the name server virtual machine.

ON
: Specifies that the name server should take a dump if the abend handler is entered.

OFF
: Specifies that the name server should NOT take a dump if the abend handler is entered.

NOW
: Specifies that the name server should take a dump right now. Execution continues after the dump is taken.

Rebuilding the Name Server Modules

To compile the name server ASMSQL and CSQL files, you must have a Database Administrator issue the following command:

```
grant connect to server_id identified by sqldbapw
```

where *server_id* is the user ID of the name server virtual machine.

If you do not want to use the password SQLDBAPW, you must issue the GRANT command with a password of your choice and modify the TCPOBJCT EXEC to reflect the new password.

Chapter 10. Configuring the BOOTP Server

The BOOTP server (daemon) responds to client requests for boot information using data maintained in a BOOTP machine file. This data includes the IP address of the client, the IP address of the TFTP daemon and information about the files to request from the TFTP daemon.

To configure the BOOTP server, you must perform the following steps:

BOOTP Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the BOOTP server.
3. Customize the ETC BOOTPTAB file.

Dynamic Server Operation

The BOOTP server provides a console subcommand interface that allows you to perform various server administration tasks. For more information see "Dynamic Server Operation" on page 258.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  BOOTPD  0
```

The BOOTP server requires that port UDP 67 be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
PORT
  67  UDP BOOTPD  ; BOOTP Server
```

Step 2: Update the DTCPARMS File

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

```
:Nick.BOOTPD
  :Parms.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

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

- Change the parameters passed to the BOOTPD command.

If no parameters are specified on the **:Parms.** tag of the DTCPARMS file, the BOOTP server is initialized with the following parameters:

```
MACHINE ETC BOOTPTAB *
```

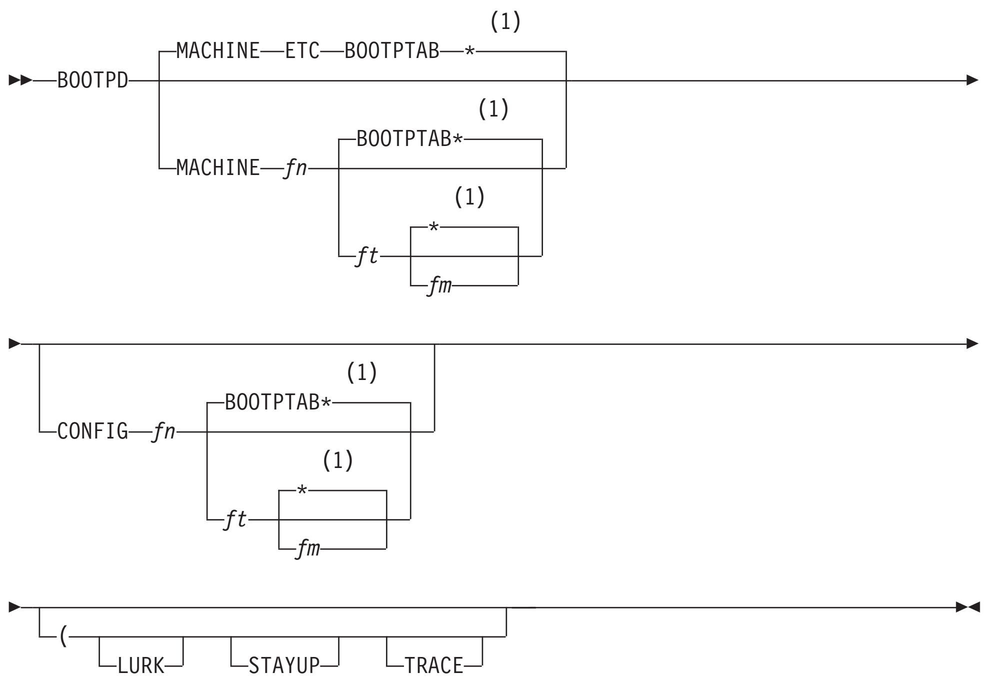
Step 3: Customize the ETC BOOTPTAB File

The BOOTP server searches the ETC BOOTPTAB file for information when it attempts to satisfy BOOT requests generated by BOOTP clients on the network. Before you use the BOOTP server, you need to customize the ETC BOOTPTAB file with information about the BOOTP clients (such as IBM Network Stations) that will be used in your environment.

A sample configuration file is provided as BOOTP SAMPLE on the TCPMAINT 591 disk. Your customized ETC BOOTPTAB file should be copied to the TCPMAINT 198 minidisk as ETC BOOTPTAB. See the comments within the ETC BOOTPTAB file for detailed information about how to specify entries within this file.

BOOTPD Command

BOOTP services are initiated using the BOOTPD command:



Notes:

1 If "*" is specified (or default) for the file mode, then the first file matching the file name and file type is used.

Specify BOOTPD command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

MACHINE
: Indicates that the file specification that follows specifies a file containing client information.

fn The file name of the file to load.

ft The file type of the file to load.

fm The file mode of the file to load.

CONFIG
: Indicates that the file specification that follows specifies a file containing configuration information. This information lists adapters on the host which should be monitored and those that should not. Also, whether forwarding of BOOT requests should occur and when and where they should be sent.

LURK
: Indicates that the BOOTP daemon should never respond to a client request. It should only listen.

STAYUP
: Indicates that the BOOTP daemon should continue to operate across VM TCP/IP failures.

TRACE
: Indicates that the BOOTP daemon should display debug information as requests are processed.

Usage Notes

- MACHINE and CONFIG are reserved keywords. They may not be used as file names or file types.
- The defaults for the BOOTPD command when no operands or options are specified are:
  - "ETC BOOTPTAB *" is the machine file,
  - LURK mode is disabled,
  - STAYUP mode is disabled,
  - TRACE mode is disabled,
  - No configuration file is used; thus, the BOOTP daemon will listen to BOOT requests received on any IP address.
- The configuration file is composed of blank lines, comment lines, and configuration statements. Configuration statements have the same function and syntax as the BOOTPD **EXCLUDE**, **FORWARD**, and **INCLUDE** subcommands.

  The format of the configuration file is:
  - One line per statement.
  - Blank lines are ignored.
  - Comment lines are ignored. Comment lines are lines where the first non-blank character is an asterisk (*) or pound symbol (#)..
- The machine file is composed of blank lines, comment lines, entry lines for the clients, and tag control lines used to decipher the entry lines. The format and content of this file is described further by comments within the BOOTPTAB SAMPLE file.
- A machine file (table) must exist for the BOOTP server to initialize and operate. This file is required because it provides the BOOTP daemon with information necessary to satisfy client BOOT requests.

However, there are cases when a BOOTP daemon may not need a *functional* machine file. One example is the case when the BOOTP server in question acts as a gateway and forwards all requests to another server. In this case, a machine file which contains only blank lines or comment lines may be used for the gateway server.

- **STAYUP** is needed only when the TCP/IP stack machine does not contain an AUTOLOG entry for the virtual machine running BOOTP.

Dynamic Server Operation

Some configuration attributes (such as tracing and client request handling) can be changed during server execution using the BOOTPD subcommands described in the next section. In addition, certain server activities can be queried or changed by subcommands. Any subcommand not understood by the BOOTP server is assumed to be a CMS command and is passed to the CMS command line for execution.

Issue BOOTPD subcommands at the BOOTP server console.

BOOTPD Subcommands

The BOOTPD subcommands are listed in Table 15. This table provides the shortest abbreviation, a description, and a

page reference for more information for each BOOTPD subcommand.

Table 15. BOOTPD Subcommands

Subcommand	Minimum Abbreviation	Description	Page
CMS	CMS	Passes a command to CMS for execution.	258
CONFIG	CONFIG	Displays configuration information.	259
EXCLUDE	EXCLUDE	Identifies adapter addresses for which BOOT requests should be ignored.	261
EXIT	EXIT	Stops the BOOTPD server and its processing. EXIT is equivalent to QUIT and STOP.	262
FORWARD	FORWARD	Controls the forwarding of BOOT requests to another BOOTPD server.	262
HELP	HELP	Displays a summary of BOOTPD subcommands.	265
INCLUDE	INCLUDE	Identifies adapter addresses for which BOOT requests should be handled.	265
LURK	LURK	Toggles the LURK mode of the BOOTPD server.	266
QUIT	QUIT	Stops the BOOTPD server and its processing. QUIT is equivalent to EXIT and STOP.	266
RELOAD	RELOAD	Reloads BOOTPD machine and configuration files.	266
STAYUP	STAYUP	Toggles the STAYUP mode of the BOOTPD server.	268
STOP	STOP	Stops the BOOTPD server and its processing. STOP is equivalent to EXIT and QUIT.	268
TRACE	TRACE	Toggles the TRACE mode of the BOOTPD server.	269

CMS Subcommand

Use the CMS subcommand to issue a command to CMS.

```
►►──┬─────┬──cms_command──────────────────────────────────────────────►◄
    └─CMS─┘
```

Operands

cms_command
The CMS command to be issued.

Usage Notes

- Do not issue any CMS command that would take considerable time to execute (for example, XEDIT). While the CMS command executes, the server does not respond to requests.
- The **CMS** keyword is usually not required because the daemon will pass any command string that is not recognized as a BOOTPD subcommand to CMS. The **CMS** keyword is used to identify CMS commands which would normally be interpreted as a BOOTPD subcommand, for example, **TRACE**.
- After completion of any command, the following ready prompt is displayed:

  ```
  BOOTPD Ready;
  ```

  or

  ```
  BOOTPD Ready (rc);
  ```

  if the return code (rc) is not zero.

CONFIG Subcommand

Use the CONFIG subcommand to display configuration information.

```
►►──CONFIG────────────────────────────────────────────────────────────►◄
```

Usage Notes

- This subcommand causes the BOOTP daemon to query the current TCP/IP configuration of the host where the BOOTP daemon is running, to determine the IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand's operation, whether or not the subcommand completes successfully.
- This subcommand produces a multiple line response which indicates the status of settings, table files used, included and excluded adapter addresses, and forwards that are active or could be activated. This output is discussed below, in sections, for clarity of meaning.

```
Lurk=l Stayup=s Trace=t
Machine Table=filespec
Configuration Table=filespec
```

This section indicates the status of the LURK, STAYUP, and TRACE settings, along with the file specifications for the machine and configuration table files used by the server.

l is 1 if LURK mode is on; 0 if LURK mode is off.

s is 1 if STAYUP mode is on; 0 if STAYUP mode is off.

t is 1 if TRACE mode is on; 0 if TRACE mode is off.

filespec
is the file name, file type and file mode of the file that is in use.

```
Included Addresses:
  Adapter adpaddr reqtype
```

This section indicates the IP addresses of adapters for which the BOOTP daemon will process requests. One “Adapter” line is displayed for each adapter that the BOOTP daemon will handle. If there are no included addresses, “NONE” is displayed instead of the “Adapter” line(s).

adpaddr
is the IP address of an adapter on the host system.

reqtype
is the type of request that will be handled. Possible values are:

CLIENTS	for BOOT requests broadcast by clients to the host.
GATEWAYS	for BOOT requests forwarded by a BOOTP daemon on behalf of a client.
ANY	for any client or gateway forwarded requests.

```
Excluded Addresses:
  Adapter adpaddr reqtype
```

This section indicates the IP addresses of adapters the BOOTP daemon should ignore. If there are no excluded addresses, “NONE” is displayed instead of the “Adapter” line(s).

adpaddr
is the IP address of the adapter.

reqtype
is the type of request that will be excluded. Possible values are:

CLIENTS	for BOOT requests broadcast by clients to the host.
GATEWAYS	for BOOT requests forwarded by a BOOTP daemon on behalf of a client.
ANY	for any client or gateway forwarded requests.

```
Forwards:
  Adapter adpaddr -> toaddr frequency actflag
  Gateway gateaddr -> toaddr frequency
```

This section indicates whether BOOT request forwarding has been specified for:

– requests received on specific adapters, or
– requests forwarded by a specific gateway.

adpaddr
is the IP address of the adapter.

gateaddr
: is a gateway IP address. The gateway IP address is the address of an adapter on which a request is initially received, but is then forwarded.

toaddr
: is the IP address of a host that is running another BOOTP daemon which should receive forwarded requests.

frequency
: is an indication of when forwarding should occur for the specified adapter or gateway. This value can be either:

ALWAYS	indicating that any request on the adapter or forwarded by the gateway should always be forwarded.
UNKNOWN	indicating that forwarding should occur only for requests on the given adapter, or forwarded by a gateway, when they cannot be handled using this daemon's machine file.

actflag
: indicates the status of the adapter for which the forward has been specified. This value can be either:

INCLUDED	indicating that the adapter is included in the configuration and handles both client and gateway-forwarded requests.
EXCLUDED	indicating that the adapter is excluded from the configuration (for example, no forwarding will occur until the adapter is included in the configuration).
PARTIAL	indicating that some BOOT requests received on the adapter will not be handled. This can occur if the **EXCLUDE** or **INCLUDE** subcommand resulted in some BOOT requests not being handled. For example, gateway forwarded requests received over a specific adapter may be excluded, while client requests may be included. For more information about how to control request handling see "INCLUDE Subcommand" on page 265 and "EXCLUDE Subcommand".

EXCLUDE Subcommand

Use the EXCLUDE subcommand to specify an adapter address to ignore. You can specify additional operands to indicate whether all requests received across a specific adapter should be ignored, or whether only client BOOT requests or gateway-forwarded requests should be ignored.

```
                                      ┌─ANY──────┐
►►──EXCLUDE──┬─ADApter──ipaddr─┬──────┼──────────┼─────────────────────────────►◄
             ├─SUBnet──netaddr─┤      ├─CLIents──┤
             └─ALL─────────────┘      ├─GATeways─┤
                                      └─ANY──────┘
```

Operands

ADApter *ipaddr*
: The IP address of the adapter on the host system that should be ignored.

SUBNET *netaddr*
The address of a subnet that should be ignored. Any host adapter address that is part of the specified subnet is ignored; *netaddr* may be any address valid for the subnet.

ALL
Indicates that all adapters should be ignored.

ANY
Indicates that any request that is received on the specified adapters (or adapters associated with a subnet, or "ALL") should be ignored. This is the default.

CLIents
Indicates that only client BOOT requests that are received on the specified adapters (or adapters associated with a subnet, or "ALL") should be ignored.

GATeways
Indicates that gateway-forwarded requests that are received on the specified adapters (or adapters associated with a subnet, or "ALL") should be ignored.

Usage Notes

- The opposite of the EXCLUDE subcommand is the **INCLUDE** subcommand. Any values set by the EXCLUDE subcommand may be reset by the INCLUDE subcommand.
- This subcommand causes the BOOTP daemon to query the current TCP/IP configuration of the host where the BOOTP daemon is running, to determine the IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand's operation, whether or not the subcommand completes successfully.

EXIT Subcommand

Use the EXIT subcommand to stop the BOOTP daemon. This subcommand is equivalent to the **QUIT** and **STOP** subcommands.

```
►►──EXIT──────────────────────────────────────────────────────►◄
```

Operands

The EXIT subcommand has no operands.

FORWARD Subcommand

Use the FORWARD subcommand to specify BOOT requests that should be forwarded to another BOOTP daemon at another IP address. Requests are selected based upon the adapter on which they are received or the gateway from which they were forwarded.

```
                                                            ┌─ALWAYS──┐
►►──FORWARD───┬─ADApter──ipaddr───┬──┬──────┬──toipaddr──┼─────────┼──────────►◄
              ├─SUBnet──netaddr───┤  └─TO───┘             ├─ALWays──┤
              ├─GATeway──gateaddr─┤                        ├─UNKnown─┤
              └─ALL───────────────┘                        └─NEVer───┘
```

Operands

ADApter *ipaddr*
: Indicates that BOOT requests received over the specified IP address should be forwarded. The IP address is the address of an adapter on the host system that would receive the request.

SUBnet *netaddr*
: Indicates that BOOT requests received over the IP addresses that are part of the specified subnet should be forwarded; *netaddr* may be any address valid for the subnet.

GATeway *gateaddr*
: Indicates that BOOT requests that were forwarded by a gateway at the specified IP address should be forwarded.

ALL
: Indicates that all IP adapter addresses should be handled as forwarding addresses.

TO *toipaddr*
: Indicates the IP address to which the BOOT request should be forwarded.

ALWAYS
: Indicates that BOOT requests that pass the selection criteria (for example, on the specified adapter or from the specified gateway) should always be forwarded. This is the default.

UNKNOWN
: Indicates that BOOT requests that pass the selection criteria (for example, on the specified adapter or from the specified gateway) but cannot be handled using this daemon's machine file should be forwarded. Requests for clients that are in the machine file are not forwarded.

NEVER
: Indicates that BOOT requests that pass the selection criteria (for example, on the specified adapter or from the specified gateway) should never be forwarded. This cancels the forwarding that may have been previously specified for the BOOT requests that match the criteria.

Usage Notes

- Forwarding applies only to requests that are not excluded by the **EXCLUDE** subcommand.
- Forwarding specified for a gateway takes precedence over forwarding specified for an adapter.
- A hop count is maintained in the BOOT request. This hop count is incremented each time a BOOTP daemon forwards the request. BOOTPD will not forward a request whose hop count is three or more.
- The BOOT request contains a server name field. This field allows the client to specify the host name of a BOOTP daemon which should process the request. If

the target BOOTP daemon is not the receiving server, and the address of the server is not on the same cable as the adapter that received the request, then the request will be forwarded.

Normally, a request is not forwarded to a target BOOTP daemon when the target daemon is on the same cable as the adapter of the BOOTP daemon that receives the original request. Such forwarding is not done because it's assumed that the target daemon would have heard this same request.

However, you can override this and force a request to be sent to such a target daemon. If a **FORWARD** to a specific IP address is defined for a receiving adapter, requests will always be forwarded to the BOOTP daemon at that IP address.

- Requests that are forwarded by a gateway contain a gateway IP address. The BOOTP server specifies this gateway IP address as the address of the adapter which receives the original request. This allows the BOOTP daemon, which ultimately builds the response packet for the requesting client, to send that packet to the correct gateway; that gateway then sends the response packet to the client.

  Only the BOOTP daemon that initially hears the client request acts as the gateway; subsequent forwarding of the request by other BOOTP daemons does not change the gateway IP address.
- Forwarding is useful if you wish to centralize your machine files on a specific host and have other BOOTP daemons forward their received requests to the central site for processing. When you set up forwarding in this manner, you must take into account the increased load on the central server and the time required to forward requests. If the interval to respond to a client request is too long, that client may then retransmit its requests, and increase the network load.

  The following is a simple example of forwarding to central sites. The configuration consists of four hosts that run VM BOOTP daemons:

  VMSAT1, VMSAT2
  Satellite VM hosts running BOOTP daemons connected to subnets with clients that submit BOOT requests.

  VMMAIN Main VM host running BOOTP for responding to BOOT requests from VMSAT1 and VMSAT2.

  VMCENT Central Master VM HOST containing a master machine file.

  In this example, requests received by the VMSAT1 or VMSAT2 are automatically routed to VMMAIN. This allows VMSAT1 and VMSAT2 to run BOOTP daemons which do not maintain a *functional* machine file. A **FORWARD** statement would appear in the configuration files for VMSAT1 and VMSAT2 as:

  ```
  FORWARD ALL TO xxx.xxx.xxx.xxx ALWAYS
  ```

  where *xxx.xxx.xxx.xxx* is the IP address of VMMAIN.

  Normally, all requests are satisfied by VMMAIN. If VMMAIN cannot handle a request, that request is forwarded to VMCENT, which runs a BOOTP daemon with the master table file for VMMAIN and other VMMAIN-type hosts. The **FORWARD** statement would appear in the VMMAIN configuration file as:

  ```
  FORWARD ALL TO yyy.yyy.yyy.yyy UNKNOWN
  ```

  where *yyy.yyy.yyy.yyy* is the IP address of VMCENT.
- This subcommand causes the BOOTP daemon to query the current TCP/IP configuration of the host where the BOOTP daemon is running, to determine the

IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand's operation, whether or not the subcommand completes successfully.

HELP Subcommand

Use the HELP subcommand to display a brief description of available subcommands.

```
►►──HELP──────────────────────────────────────────────────────────────►◄
```

Operands

The HELP subcommand has no operands.

INCLUDE Subcommand

Use the INCLUDE subcommand to specify the adapter address or address of BOOTP forwarding daemons for which requests should be handled.

```
                                         ┌─ANY──────┐
►►──INCLUDE──┬─ADApter──ipaddr─┬──┼──────────┼──────────────────────►◄
             ├─SUBnet──netaddr─┤  ├─CLIents──┤
             └─ALL─────────────┘  ├─GATeways─┤
                                  └─ANY──────┘
```

Operands

ADApter *ipaddr*
: The IP address of the adapter on the host system for which requests should be handled.

SUBnet *netaddr*
: The address of a subnet for which requests should be handled. Any host adapter address that is part of the specified subnet is handled; *netaddr* may be any address valid for the subnet.

ALL
: Indicates that requests from all adapters should be handled.

ANY
: Indicates that any request that is received on the specified adapters (or adapters associated with a subnet, or "ALL") should be handled. This is the default.

CLIents
: Indicates that only client BOOT requests that are received on the specified adapters (or adapters associated with a subnet, or "ALL") should be handled.

GATeways
: Indicates that gateway-forwarded requests that are received on the specified adapters (or adapters associated with a subnet, or "ALL") should be handled.

Usage Notes

- The opposite of the INCLUDE subcommand is the **EXCLUDE** subcommand. Any values set by the INCLUDE subcommand may be reset by the EXCLUDE subcommand.
- This subcommand causes the BOOTP daemon to query the current TCP/IP configuration of the host where the BOOTP daemon is running, to determine the IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand's operation, whether or not the subcommand completes successfully.

LURK Subcommand

Use the LURK subcommand to toggle the LURK mode in the BOOTP daemon. If the daemon is already operating in LURK mode, it will resume answering requests from clients. If the daemon is not operating in LURK mode, it will begin to listen for, but not will not respond to, client requests.

```
►►──LURK──────────────────────────────────────────────────────────────►◄
```

Operands

The LURK subcommand has no operands.

Usage Notes

- The following is displayed upon completion of this subcommand:

  ```
  LURK is now l
  ```

  where *l* is 0 if LURK mode is off; 1 if LURK mode is on.

QUIT Subcommand

Use the QUIT subcommand to stop the BOOTP daemon. This subcommand is equivalent to the **EXIT** and **STOP** subcommands.

```
►►──QUIT──────────────────────────────────────────────────────────────►◄
```

Operands

The QUIT subcommand has no operands.

RELOAD Subcommand

Use the RELOAD subcommand to reload the machine or configuration table.

```
                                         (1)
                 ┌─MACHINE──ETC──BOOTPTAB──*──────────────┐
►►──RELOAD───────┼────────────────────────────────────────┼──────────►◄
                 │                      (1)               │
                 │           ┌─ETCBOOTPTAB*─────────┐     │
                 ├─MACHINE───┼──────────────────────┼─────┤
                 │           │                 (1)  │     │
                 │           │      ┌─BOOTPTAB *─┐  │     │
                 │           └─filename─┼────────┼──┘     │
                 │                  │       (1)  │        │
                 │                  │      ┌─*─┐ │        │
                 │                  └─filetype─┼───┼─┘    │
                 │                         └─filemode─┘   │
                 │                     (1)                │
                 │          ┌─CNFBOOTPTAB*────────┐       │
                 └─CONFIG───┼─────────────────────┼───────┘
                            │                (1)  │
                            │     ┌─BOOTPTAB *─┐  │
                            └─filename─┼───────┼──┘
                                  │        (1) │
                                  │       ┌─*─┐│
                                  └─filetype─┼───┼┘
                                          └─filemode─┘
```

Notes:

1 If '*' is specified (or default) for the file mode, then the first file matching the file name and file type is used.

Operands

MACHINE
Indicates that the file to be loaded is a machine file that contains client information.

CONFIG
Indicates that the file to be loaded is a configuration file.

filename
The file name of the machine or configuration file to load.

filetype
The file type of the file to load.

filemode
The file mode of the file to load.

Usage Notes

- MACHINE and CONFIG are reserved keywords. They may not be used as file names or file types.
- The configuration file is composed of blank lines, comment lines, and configuration statements. Configuration statements have the same function and syntax as the BOOTPD **EXCLUDE**, **FORWARD**, and **INCLUDE** subcommands.

  The format of the configuration file is:
  - One line per statement.
  - Blank lines are ignored.
  - Comment lines are ignored. Comment lines are lines where the first non-blank character is an asterisk (*) or pound symbol (#).

- The machine file is composed of blank lines, comment lines, entry lines for the clients, and tag control lines used to decipher the entry lines. The format and content of this file is described further by comments within the BOOTPTAB SAMPLE file.
- When a configuration file is processed, all adapters are assumed to be included and no forwarding exists until specified otherwise by configuration file statements.
- If the machine file is successfully loaded, the following response is displayed:

  ```
  fn ft fm  loaded in secs seconds.
  Table reloaded from fn ft fm
  BOOTPD Ready;
  ```

  where *fn*, *ft*, and *fm* are the respective file name, file type and file mode of the loaded file, and *secs* is the time required to load the file.

STAYUP Subcommand

Use the STAYUP subcommand to toggle the STAYUP mode in the BOOTP daemon. If the daemon is already operating in STAYUP mode, it will cease operating in this mode and will end processing if a subsequent TCP/IP failure occurs. If the daemon is not operating in STAYUP mode, it will begin to ensure processing will not end if a subsequent TCP/IP failure occurs.

```
►►──STAYUP──────────────────────────────────────────────────────►◄
```

Operands

The STAYUP subcommand has no operands.

Usage Notes

- This subcommand is needed only when the TCP/IP stack machine does not contain an AUTOLOG entry for the virtual machine running BOOTPD.
- The following is displayed upon completion of the subcommand:

  ```
  STAYUP is now s
  ```

  where *s* is 0 if STAYUP mode is off; 1 if STAYUP mode is on.

STOP Subcommand

Use the STOP subcommand to stop the BOOTP daemon. This subcommand is equivalent to the **EXIT** and **QUIT** subcommands.

```
►►──STOP────────────────────────────────────────────────────────►◄
```

Operands

The STOP subcommand has no operands.

TRACE Subcommand

Use the TRACE subcommand to toggle the TRACE mode in the BOOTP daemon. If the daemon is already operating in TRACE mode, it will cease displaying debug information as it processes requests. If the daemon is not operating in TRACE mode, it will display debug information as it processes requests.

```
►►──TRACE──────────────────────────────────────────────────────────────►◄
```

Operands

The TRACE subcommand has no operands.

Usage Notes

- The following is displayed upon completion of this subcommand:
  ```
  TRACE is now t
  ```
  where *t* is 0 if TRACE mode is off; 1 if TRACE mode is on.
- See the *TCP/IP Diagnosis Guide* for a description of BOOTP server trace output.

Chapter 11. Configuring the DHCP Server

The DHCP daemon (the DHCPD server) responds to client requests for boot information using information contained in a DHCP machine file. This information includes the IP address of the client, the IP address of the TFTP daemon and information about the files to request from the TFTP daemon.

Unlike the BootP daemon, each client (with its assigned IP address) that is serviced by the DCHP deamon does not need to be listed in a machine file. IP addresses may be allocated:

Dynamically	An IP address is temporarily assigned to the client for a specified period of time. The "lease" on the address must be renewed prior to the end of the specified lease if the client wishes to keep using the IP address.
Automatically	An IP address is permanently assigned to a client from the list of available addresses.
Manually	A specific IP address (specified in the machine file) is assigned to the client.

To configure the DHCPD virtual machine, you must perform the following steps:

DHCPD Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for DHCPD.
3. Configure the ETC DHCPTAB file.
4. Construct a DHCPD machine file.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
DHCPD  0
```

The DHCPD server listens on port 67. Verify that the following statement is added to your TCPIP server configuration file as well:

```
PORT
67  UDP DHCPD  ; DHCPD; Server
```

The DHCPD; server uses raw IP functions. Verify that the following statement is added to your TCPIP server configuration file:

```
OBEY
  DHCPD
ENDOBEY
```

Step 2: Update the DTCPARMS File for DHCPD

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

```
:Nick. DHCPD
  :Parms.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the DHCP server if you change the parameters passed to the DHCPD command.

If no parameters are specified on the **:Parms.** tag in the DTCPARMS file, the DHCP server is initialized with the following parameters:

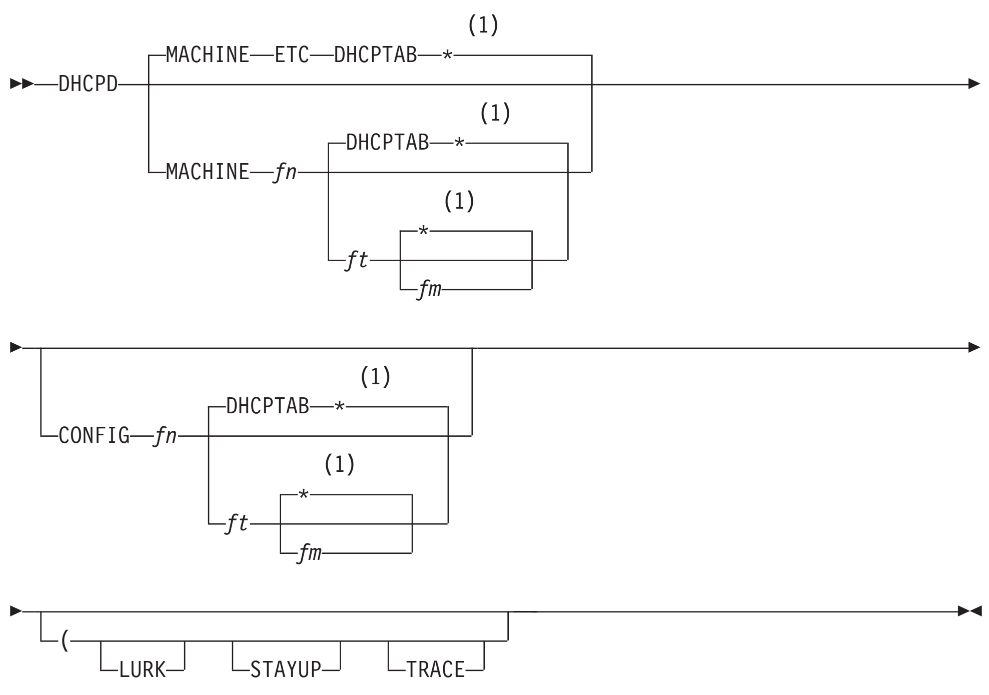
```
MACHINE ETC DHCPTAB *
```

Step 3: Configure the ETC DHCPTAB File

The DHCPD server searches the ETC DHCPTAB file for information when it attempts to satisfy DHCP requests generated by BootP or DHCP clients on the network. Before you use the DHCPD; server, you need to customize the ETC DHCPTAB file with information about the IBM Network Stations that will be used in your environment. See "Step 4: Construct a DHCPD Machine File" on page 277 for detailed information about how to specify entries within this file. The customized ETC DHCPTAB file should remain on the TCPMAINT 198 minidisk.

DHCPD Command

DHCP services are initiated using the DHCPD command:



Notes:

1 If "*" is specified (or defaulted) for the file mode, the first file that matches the given file name and file type is used.

Specify DHCPD command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

MACHINE
: indicates that the file specification that follows specifies a file containing client information.

fn is the file name of the file to load.

ft is the file type of the file to load.

fm is the file mode of the file to load.

CONFIG
: indicates that the file specification that follows specifies a file containing configuration information. This information lists adapters on the host which should be monitored and those that should not. Also, whether forwarding of BootP/DHCP requests should occur, and when and where they should be sent.

LURK
: indicates that the DHCP daemon should never respond to a client request. It should only listen.

STAYUP
: indicates that the DHCP daemon should continue to operate across VM TCP/IP failures.

TRACE
indicates that the DHCP daemon should display debug information as requests are processed.

Usage Notes

- The DHCP server saves information on the addresses that are in use and the clients that are using them in a file on the daemon’s A-disk. This file, DHCPD BINDINFO, is read at server initialization and is updated as the status of supported clients and addresses changes over time.

  A work file, DHCPDWRK BINDINFO, is maintained in addition to the DHCPD BINDINFO file. Neither of these files should be changed by the user.
- MACHINE and CONFIG are reserved keywords. They may not be used as file names or file types.
- The defaults for the DHCPD command when no operands or options are specified are:
  - "ETC DHCPTAB *" is the machine file,
  - LURK mode is disabled,
  - STAYUP mode is disabled,
  - TRACE mode is disabled,
  - No configuration file is used; thus, the DHCP daemon will listen for BootP and DHCP requests received on any IP address.
- The configuration file is composed of blank lines, comment lines, and configuration statements. Configuration statements have the same function and syntax as the DHCPD EXCLUDE, FORWARD, and INCLUDE subcommands.

  The format of the configuration file is:
  - One line per statement.
  - Blank lines are ignored.
  - Comment lines are ignored. Comment lines are lines where column one is an asterisk (*) or pound symbol (#).
- The machine file is composed of blank lines, comment lines, entry lines for the clients, and tag control lines used to decipher the entry lines. The format and content of this file is described further in the section “DHCPD Machine Statements” on page 286.
- A machine file (table) must exist for the DHCP server to initialize and operate. This file is required because it provides the DHCP daemon with information necessary to satisfy client BootP/DHCP requests.

  However, there are cases when a DHCP daemon may not need a *functional* machine file. For example, when the DHCP server in question acts as a gateway and forwards all requests to another server. Here, a machine file which contains only blank lines or comment lines may be used for the gateway server.
- STAYUP is needed only when the TCP/IP machine does not contain an entry for the virtual machine running DHCPD.

Machine File Overview

The DHCP server provides machine information to clients based on statements contained in the server’s machine file and based on information provided by the client. The server’s machine 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, you must have created the machine file. Once the DHCP server program is running, you can also make dynamic changes to the configuration by modifying the machine file and using the RELOAD subcommand to update the machine information. For more information on the RELOAD subcommand, see “RELOAD Subcommand” on page 329.

You create a hierarchy of machine parameters for a DHCP-supported 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. The task of configuring subnets also requires you to set lease time and other options for clients using the subnet.
- 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 that all use a shared printer could comprise a Class.
- 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.

The concept of scoped statements in DHCP machine file is shown by the following:

```
option A 1
option B 2
option C 3

client k
{
}

subnet y
{
  client m
  {
    option E 11
    option H 12
  }

  class Q
  {
    option G
    option F 9
  }
  option C 6
  option E 5
}

subnet z
{
  option F 7
}

class x
{
  option E 1
}
```

In this example:

- Options A, B, and C are global and are inherited by all clients in the network unless overridden by a value for the same option at a lower level in the network. Client K is a specific client specified at the global level.
- Clients that are specifically defined under a subnet, such as client M, must be located in that subnet in order to be served. Client M must indicate it is in subnet Y in order to be served.
- Clients that are not specifically defined will automatically fall into a specific location in the hierarchy based on their current network location.
- Option G is served only to clients that belong to class Q in subnet Y.
- Option E, with a value of 11, and option H, with a value of 12, are served only to client M in subnet Y.
- Option C is defined in two places. For all clients in subnet Y, the value of option C is 6. For all other clients, the value of option C is 3.
- Option F is also defined in two places. For all clients of class Q in subnet Y, the value of option F is 9. For all clients in subnet Z, the value of option F is 7. For other clients, option F is not defined.
- Class X options are served to any client that requests class X. Class Q options are served only to clients in subnet Y that request class Q.
- Option E is defined in three places. For clients not in subnet Y that request to be in class X, the value of option E is defined as the value 1. For the remaining clients in subnet Y except client M, the value of option E is 5 because a Subnet statement is considered more specific than a global Class statement. For client M, the value of option E is 11, because a CLIENT statement is considered more specific than a Subnet statement.
- For clients in subnet Z that do not request class X, option E is undefined.

Statement Precedence Order

As the server searches the machine file for statements related to the client request, it constructs an order of precedence that is used to determine which parameter to select when a parameter is specified at more than one level.

- Statements which are most specific have the highest precedence.
- 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-scoped level.
- Parameters specified within braces under a statement, such as a Subnet statement, are considered local and apply only to clients within the hierarchical level defined by that statement.
- Parameters scoped at a client level take precedence over all other parameters at that level (global or subnet).

- Parameters in a subnet take precedence over parameters in a class unless the class is scoped within the subnet.
- Parameters scoped at a class level take precedence over global parameters.
  - If a client is specified at both a global level and a subnet level, then the subnet level client parameters take precedence over the global level client parameters.
  - If a class is specified at both a global level and a subnet level, then the subnet level class parameters take precedence over the global level class parameters.
- Vendor statements always have a global scope.

For example, if option 1 is specified as both a class option at the global level and as a subnet level option, then the class level option is used because it is more specific to the client's environment.

Step 4: Construct a DHCPD Machine File

The machine file defines the information that will be returned to clients as configuration parameters and determines how addresses are to be assigned. It is important for you to layout the information in the simplest manner possible. This section will walk you through the construction of a machine file in a step by step manner. For the sake of a robust example, we will override most of the statement defaults in order to discuss them and construct a large sample machine file. In your case, you will probably be able to rely on the defaults.

Global Information

Certain information is global to all clients that are being serviced or relates to the overall operation of the server. This information should be determined first and placed at the global level.

Ping Time

Before assigning an IP address to a client for the first time, DHCPD will issue an ICMP Echo Request to that address (in other words, PING it). The amount of time it will wait for a response before assigning that address is determined by the PingTime statement. If the DHCPD server receives a response from a machine actively using the address, then it knows that there is a problem and it cannot reissue the address. The default is one second. If more time is needed within your environment for a ping response to reach your DHCPD server, then add the PingTime statement.

One way to determine whether your installation will need more time is to use the TCP/IP PING command to sample the ping response times for the addresses that will be supported. By choosing addresses of active machines already on the target subnets you can determine how long it takes them to respond and thereby determine an appropriate time interval.

For our example, the default time of 1 second is too small so a new PingTime of 2 seconds is set.

```
PingTime 2 SEC
```

Reserved Address Time

If the client requests boot parameters using DHCP protocol, then it expects that it could receive multiple offers of addresses if there is more than one DHCP server listening. Generally, the client will wait a specified period of time to receive all DHCP offers, then select one. Because there is the possibility that another server might be selected *and* our DHCPD server might not see the accept packet (because of an error somewhere), DCHP needs to know how long to reserve an IP address before putting it back into the address pool for reuse. The ReservedTime statement lets you change the time the server will wait before giving up on an offer and reusing the address. In our case, assume that our clients have been set to respond to an offer in one minute instead of the default of five minutes.

```
ReservedTime 1 minute
```

Setting the Lease Duration

DHCPD can reserve addresses either indefinitely or for a particular lease time. Indefinite leases are used primarily for BootP clients, because these clients do not support the concept of leasing an address for a specified period of time. Specific lease intervals are used for most DHCP clients. The clients are provided an address for a specified period of time, and may extend the lease during their lease time.

For our example, the default lease period of 24 hours is too long. We have a limited number of addresses that must be reused as one shift of users leaves for the day, and another takes over, using different hardware. Therefore, we want a lease time default of 9 hours. The LeaseTimeDefault statement allows us to set that lease interval.

```
LeaseTimeDefault 9 hours
```

Setting Wait Time Before Reusing an Expired Lease

When a lease on an address expires, it is best to let the server wait a period of time before returning the address to the address pool, because it is possible that the client might have been delayed in requesting to extend the lease. The UsedIPAddressExpireInterval statement lets you tailor how long the server will wait before returning an address to the address pool for reuse. For our example, the default of 30 seconds is too short; double it to one minute.

```
UsedIPAddressExpireInterval 1 minute
```

Setting How Often to Check Addresses

DHCPD does not maintain individual timers for each IP address that it has allocated. Instead, it periodically checks the status of all addresses that are being leased. This checking covers IP addresses with active leases, reserved leases and IP addresses whose leases have expired and are waiting to return to the address pool. The default interval for checking lease statuses is one minute. For our example, assume that we do not want the inherent overhead of checking leases every minute and instead use 15 minutes as the interval. Since we only check the leases every 15 minutes, it is possible for a lease to expire at 1 minute into the interval and we will not handle it until the check that is done 14 minutes later. With our example server, this is not a problem; we are going to accept a delay in handling leases and obtain a benefit of less virtual processor usage. The LeaseExpireInterval statement lets us set this interval.

```
LeaseExpireInterval 15 seconds
```

Supporting BootP Requests

By default, DHCPD supports BootP requests. You may at some time choose that only DHCP requests are supported, so all addresses are served as leased addresses instead of allowing BootP clients to obtain indefinite leases on addresses. The SupportBootP statement lets you control whether BootP requests are supported. In order to aid in maintenance of our machine file, we will add the statement so that anyone looking at the file at a later time will easily know that BootP requests are supported by the server.

```
SupportBootP yes
```

Supporting Requests from Unlisted Clients

By default, DHCPD will support any client which submits a request. You may wish to require that the client be specified in the machine file in order to be served an address. In our example, we will specify the SupportUnlistedClients statement with the "no" operand.

```
SupportUnlistedClients no
```

Specifying the Next Server to Use in the Boot Process

The clients that receive the response from the DHCP server need to know the IP address of the next server to contact in the boot process. Normally, this is the address of a TFTP server that would deliver the boot image to the client.

Use the BootStrapServer statement to specify the IP address of the next server. This statement may be specified at the global, class, subnet or client levels. For our example, all clients are served by the same server, so specify the information at the global level.

```
BootStrapServer 9.100.40.75
```

Defining New Configuration Options

Over time, it is possible that new configuration options will be defined, and others considered. To simplify adding support for a client that requires a new option, a statement to define options is supported. In our example, one of our clients needs option 181 which is a list of IP addresses for machines that receive datagrams from the client machine. Since the server does not recognize option 181, we could code the option as a hexadecimal value,

```
 option 181 hex 0964320109643202
```

Or, we could use the DefineOptions statement to define the required option, then use this newly defined option later within the machine file.

```
DefineOptions
{
 option 181 ipalist
}
```

The DefineOptions statement must precede the first Option statement. The Option definition statements are surrounded by a line which contains a left brace ({) and a line containing a right brace (}). The braces indicate that the lines which they surround are part of the DefineOptions level.

Global Options

There will be some options that you wish to serve to all clients supported by your DHCPD server, regardless of the subnet on which they reside. These options are defined at the global level of the machine file.

In our case, all of our clients receive the address of the RFC 868 Time Server and the time offset for our area. The Option statement lets you specify the option value.

```
################################################################
# Time Server data:                                            #
#   option 2 -> offset of the time server from UTC in seconds   #
#   option 4 -> IP address of an RFC 868 time server           #
################################################################
option 2   -18000
option 4   9.100.40.75
```

All of the subnets that this server supports use the same Name Server, Domain Name and Domain Name Server, so we want to specify this information at the global level.

```
################################################################
# Options related to all subnets served by this server.        #
#   option 5  -> Name Server IP addresses                      #
#   option 6  -> Domain Name Server IP addresses               #
#   option 15 -> Domain Name                                   #
################################################################
  option 5   9.100.25.252
  option 6   9.100.25.252
  option 15  endicott.ibm.com
```

Specifying Option 43 Vendor Data

Some clients receive additional option data that is not architected by the DHCP RFCs. Only the manner in constructing the data is architected. This data is transmitted to the client using option 43. The data for the option is in the form of a one byte option number from 1 to 254, followed by a one byte field indicating the length of the data, followed by the data itself.

DHCPD provides the appropriate option 43 data to clients on a vendor-specific basis. This is done by matching option 60 (Vendor Class Identifier) data provided by the client with a vendor string, specified as part of a Vendor statement in the machine file. In our example, we have one type of client which uses vendor data; that client specifies option 60 with the string "IBMSPG 1.0.0".

```
################################################################
# Vendor "IBMSPG 1.0.0" returns option 43.                     #
################################################################
Vendor "IBMSPG 1.0.0"
{
 option 1 "DEBUG ON"
 option 2 hex 05
}
```

For our example client, we have defined two options to be returned as option 43 data. We specified option 1 as data to be translated to ASCII; option 2 as a hexadecimal value. Note that the normal rules for handling option values do not apply at the vendor level; the data is assumed to be either ASCII or hexadecimal. The left brace ({) and right brace (}) statements signify these options belong to the preceding Vendor statement (at the vendor level). We chose to code the data to be returned using two Option statements, to make this information more readily apparent. We also could have defined this data through the use of a single "option 43" statement. If this was done, we would have used the "hex" operand, and specified both options — and their associated values — as a single hexadecimal value.

Specifying Global Class Data

Some clients specify option 77 on their request. This option specifies an ASCII string which is the name of the client class; DHCPD compares this string to the values specified on Class statements. These statements indicate a level under which additional options may be specified to be returned to the client. In our example, we recognize a single class, "IBM Network Station"™. This class needs additional information to enable it to configure.

```
################################################################
# IBM Network Station manager data:                            #
#   option 67 -> Name of the boot file for the client to request#
################################################################
```

```
class "IBM Network Station"
{
    option 67  /QIBM/ProdData/NetworkStation/kernel
}
```

Specifying Global Client Data

You may need to specify options that are unique to a particular client, and do not depend on the subnet to which that client is attached. The Client statement lets you identify such clients and options that apply to them.

The Client statement that follows identifies a client machine that may be served when it is attached to any subnet supported by this server. The first two operands of the Client statement identify the client hardware type and MAC address; the third operand indicates that this client may be served any available address. The left ({) and right (}) braces then create a client level specific to this client; any option data that follows the left brace will be served to only the previously identified client. In our example, Option 12 is used to assign the host name of BIGSHOT to the client.

```
################################################################
# Globally defined Clients                                     #
################################################################
client 6 0000E5E8DC61 any
{
 option 12 BIGSHOT
}
```

Our next Client statement identifies a client machine that may be served only when it is attached to the subnet which supports address 9.100.58.240. The first two operands on the Client statement identify the client by specifying a hardware type of 0, and through an ASCII value (STEVE'S MACHINE), a client identifier; this value will be provided by the client (through the use of option 61) when it issues a boot request. The third operand indicates this client may be served only address 9.100.48.240; if this address is not already allocated, it will be reserved for this specific client. As before, the left ({) and right braces (}) are used to define client level information that is unique to the preceding client. In this case, Option 12 assigns the host name of STEVIEG to the client. Option 181 was previously defined by the DefineOptions statement and is used here.

```
client 0 "STEVE'S MACHINE" 9.100.58.240
{
 option 12 STEVIEG
 option 181 9.100.57.110
}
```

Usually, it is better to specify clients using only their hardware type and machine address, because this allows for improved performance when client information is located. Also, restricting a client to a specific hardware address removes flexibility in assigning addresses by DCHPD, and requires the DHCPD administrator to update the machine file when the machine moves to a different subnet.

Specifying the IP address to assign to a client removes some of the flexibility provided by the DHCPD server. The specified address can be used only by the client; no other client may use the address. It is recommended that you let the DHCPD server choose the address from its pool of available addresses, because this allows the possibility of address sharing. For example, one user might use an address and free it up in time for the address to be given to another user.

Subnet Related Data

Once you have determined and specified global information for the server, you now have to layout the information unique to the various subnets that the server is supporting. The Subnet statements, which we will discuss shortly, identify the subnets and the pools of addresses that they will serve. Again, left ({) and right (}) braces will be used to group Option, and other statements, with specific Subnet statements.

Our example DHCPD server will service the following environment: two floors of a building where four subnets are defined. Two of these subnets are directly attached to the machine running the VM system; activity for the other two subnets will be forwarded by BOOTPD relay agents. This example is based on an actual test environment, where only the IP addresses changed.

Table 16. DHCPD Machine File - Example Subnet Environment

Subnet Address	Subnet Information	
9.100.57.0	**Subnet Mask:**	255.255.255.0
	Routers:	9.100.57.253
	DHCP Controlled Addresses:	9.100.57.43-9.100.57.99, excluding 9.100.57.50, 9.100.57.60 and 9.100.57.85.
	Additional Requirements:	None
9.100.58.0	**Subnet Mask:**	255.255.255.0
	Routers:	9.100.58.253
	DHCP Controlled Addresses:	9.100.58.103
	Additional Requirements:	A specific machine of hardware type 6, MAC address 0000E5E78650 is never allowed to be attached to this subnet.
9.100.48.0	**Subnet Mask:**	255.255.255.0
	Routers:	9.100.48.253
	DHCP Controlled Addresses:	9.100.48.200-9.100.48.245
	Additional Requirements:	Machines of a specific class, ″IBM Network Station″, must be allocated IP addresses between 9.100.48.200 and 9.100.48.230.
9.100.176.0	**Subnet Mask:**	255.255.255.0
	Routers:	9.100.176.1
	DHCP Controlled Addresses:	9.100.176.20-9.100.176.35 9.100.176.100-9.100.176.105
	Additional Requirements:	None

For subnet 9.100.57.0, a large range of addresses have been set aside for the DHCP server to administer. Within that range of addresses there are three addresses that are still in use by non-DHCP/BootP protocol machines. We could code 4 sets of subnet statements for

- 9.100.57.43-9.100.57.49
- 9.100.57.51-9.100.57.59
- 9.100.57.61-9.100.57.84
- 9.100.57.86-9.100.57.99.

Instead it will be more straightforward to code a single subnet statement and exclude the addresses which DHCP should not handle. The subnet statement identifies the subnet address, the subnet mask, and the range of addresses that DHCP allocates for that subnet statement. Either within the subnet level or at the global level, Client statements may be specified to further restrict the range. The hardware type of zero and identifier of zero indicate that the Client statement is excluding an address.

```
################################################################
# Subnet 9.100.57.0                                            #
#   option 3  -> Router IP addresses                           #
#   option 1  -> subnet mask (This option is generated by the   #
#                SUBNET statement. It should not be specified   #
#                as an option.)                                 #
################################################################
Subnet 9.100.57.0 255.255.255.0 9.100.57.43-9.100.57.99
{
 client 0 0 9.100.57.50
 client 0 0 9.100.57.60
 client 0 0 9.100.57.85
```

Along with specifying the addresses to allocate and the subnet mask to return to the requesting clients, we need to indicate the address of a router that is used by this subnet. We will also specify a closing right brace (}) to indicate the completion of the subnet level.

```
 option 3 9.100.57.253
}
```

For subnet 9.100.58.0, only one address is available for allocation by DHCP. Also, a specific client machine may never be allowed to attach to this subnet.

```
################################################################
# Subnet 9.100.58.0                                            #
#   option 3  -> Router IP addresses                           #
#   option 1  -> subnet mask (This option is generated by the   #
#                SUBNET statement. It should not be specified   #
#                as an option.)                                 #
################################################################
Subnet 9.100.58.0 255.255.255.0 9.100.58.103-9.100.58.103
{
 option 3 9.100.58.253
 client 6 0000E5E78650 none
}
```

For subnet 9.100.48.0, 46 IP addresses are available for allocation by the server, but 31 are reserved for a specific class of machine (IBM Network Station). In this case, we only need to specify a Class statement with an IP address range, because for these machines, we previously specified a similar Class statement at the global level; that statement provides related options for all machines of this class.

```
#################################################################
# Subnet 9.100.48.0                                             #
#   option 3  -> Router IP addresses                            #
#   option 1  -> subnet mask (This option is generated by the   #
#                SUBNET statement. It should not be specified   #
#                as an option.)                                 #
#################################################################
Subnet 9.100.48.0 255.255.255.0 9.100.48.200-9.100.48.245
{
 option 3 9.100.48.253
 class "IBM Network Station" 9.100.48.200-9.100.48.230
}
```

For subnet 9.100.176.0, 22 addresses are available in two address ranges. Because the available address ranges are grouped at different ends of the total subnet address range, we will create two groups of subnet statements, instead of a single range with many Client statements to exclude unwanted addresses. We will also code, at the global level, a BALANCE: statement so that DHCPD will attempt to alternately use the two subnet statements to satisfy requests. The Balance: statement contains a label that is specified on both subnet statements, in our case "sub176". Without the balance statement, all addresses would be chosen from the first range of addresses before the second range was used.

```
Balance: sub176

#################################################################
# Subnet 9.100.176.0, addresses 20-35                           #
#   option 3  -> Router IP addresses                            #
#   option 1  -> subnet mask (This option is generated by the   #
#                SUBNET statement. It should not be specified   #
#                as an option.)                                 #
#################################################################
Subnet 9.100.176.0 255.255.255.0 9.100.176.20-9.100.176.35 label:sub176
{
 option 3 9.100.176.1
}

#################################################################
# Subnet 9.100.176.0, addresses 200-230                         #
#   option 3  -> Router IP addresses                            #
#   option 1  -> subnet mask (This option is generated by the   #
#                SUBNET statement. It should not be specified   #
#                as an option.)                                 #
#################################################################
Subnet 9.100.176.0 255.255.255.0 9.100.176.200-9.100.176.230 label:sub176
{
 option 3 9.100.176.1
}
```

This completes the machine file. The following example shows the completed file with the Balance: statement moved to the beginning of the file.

```
Balance: sub176
PingTime 2 SEC
ReservedTime 1 minute
LeaseTimeDefault 9 hours
UsedIPAddressExpireInterval 1 minute
LeaseExpireInterval 15 seconds
SupportBootP yes
SupportUnlistedClients no
BootStrapServer 9.100.40.75

DefineOptions
{
 option 181 ipalist
}
```

```
#################################################################
# Time Server data:                                             #
#   option 2 -> offset of the time server from UTC in seconds   #
#   option 4 -> IP address of an RFC 868 time server            #
#################################################################
option 2   -18000
option 4   9.100.40.75

#################################################################
# Options related to all subnets served by this server.         #
#   option 5  -> Name Server IP addresses                       #
#   option 6  -> Domain Name Server IP addresses                #
#   option 15 -> Domain Name                                    #
#################################################################
option 5   9.100.25.252
option 6   9.100.25.252
option 15  endicott.ibm.com

#################################################################
# Vendor "IBMSPG 1.0.0" returns option 43.                      #
#################################################################
Vendor "IBMSPG 1.0.0"
{
 option 1 "DEBUG ON"
 option 2 hex 05
}

#################################################################
# IBM Network Station manager data:                             #
#   option 67 -> Name of the boot file for the client to request#
#################################################################
class "IBM Network Station"
{
    option 67  /QIBM/ProdData/NetworkStation/kernel
}

#################################################################
# Globally defined Clients                                      #
#################################################################
client 6 0000E5E8DC61 any
{
 option 12 BIGSHOT
}

client 0 "STEVE'S MACHINE" 9.100.58.240
{
 option 12 STEVIEG
 option 181 9.100.57.110
}

#################################################################
# Subnet 9.100.57.0                                             #
#   option 3  -> Router IP addresses                            #
#   option 1  -> subnet mask (This option is generated by the   #
#                SUBNET statement. It should not be specified   #
#                as an option.)                                 #
#################################################################
Subnet 9.100.57.0 255.255.255.0 9.100.57.43-9.100.57.99
{
 client 0 0 9.100.57.50
 client 0 0 9.100.57.60
 client 0 0 9.100.57.85
 option 3 9.100.57.253
}

#################################################################
```

```
# Subnet 9.100.58.0                                            #
#   option 3  -> Router IP addresses                           #
#   option 1  -> subnet mask (This option is generated by the  #
#                SUBNET statement. It should not be specified  #
#                as an option.)                                #
################################################################
Subnet 9.100.58.0 255.255.255.0 9.100.58.103-9.100.58.103
{
 option 3 9.100.58.253
 client 6 0000E5E78650 none
}

################################################################
# Subnet 9.100.48.0                                            #
#   option 3  -> Router IP addresses                           #
#   option 1  -> subnet mask (This option is generated by the  #
#                SUBNET statement. It should not be specified  #
#                as an option.)                                #
################################################################
Subnet 9.100.48.0 255.255.255.0 9.100.48.200-9.100.48.245
{
 option 3 9.100.48.253
 class "IBM Network Station" 9.100.48.200-9.100.48.230
}

################################################################
# Subnet 9.100.176.0, addresses 20-35                          #
#   option 3  -> Router IP addresses                           #
#   option 1  -> subnet mask (This option is generated by the  #
#                SUBNET statement. It should not be specified  #
#                as an option.)                                #
################################################################
Subnet 9.100.176.0 255.255.255.0 9.100.176.20-9.100.176.35 label:sub176
{
 option 3 9.100.176.1
}

################################################################
# Subnet 9.100.176.0, addresses 200-230                        #
#   option 3  -> Router IP addresses                           #
#   option 1  -> subnet mask (This option is generated by the  #
#                SUBNET statement. It should not be specified  #
#                as an option.)                                #
################################################################
Subnet 9.100.176.0 255.255.255.0 9.100.176.200-9.100.176.230 label:sub176
{
 option 3 9.100.176.1
}
```

DHCPD Machine Statements

The DHCP server provides machine information to clients based on statements contained in the server's machine file and based on information provided by the client. The server's machine 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.

Machine File Format

In the DHCP machine file:

- Comments must begin with a pound sign (#), semi-colon (;) or asterisk (*) in column 1.
- Text strings that include spaces must be surrounded by single (') or double quotes (").

- Parameters to the right of a left parenthesis are ignored. They are supported for compatibility with other IBM DHCP servers.
- A continuation character (\) at the end of a line indicates the information is continued on the next line.
- Braces are used to specify statements that are scoped within other statements.
- Class statements may appear at the global level or scoped within a subnet.
- Client statements may appear at the global level or scoped within a subnet.
- Subnet statements may appear only at the global level.
- Keywords are not case sensitive. Capitalization patterns used in this documentation are not required in the configuration file.

Table 17. DHCPD Machine File Statements

Statement	Minimum Abbreviation	Description	Page
{	{	Indicates the start of a level. Statements which follow the left brace are related to the previous statement.	288
}	}	Indicates the end of a level.	289
Balance:	BALANCE:	Specifies subnet groups from which addresses should be assigned in an alternating manner.	289
BootStrapServer	BSS	Specifies the IP address of the next server to contact in the boot strap process. Normally, this will be the address of the TFTP server which delivers the boot image.	290
Class	CLASS	Identifies a user defined class (specified by the client using option 77) and a range of addresses to assign to the class.	290
Client	CLIENT	Identifies a client by an ID, which may be followed by additional options. Can also be used to identify an IP address that should not be used within a range of addresses.	291
DefineOptions	DEFOPTS	Defines (or redefines) the values allowed on Option statements.	293
InOrder:	INORDER:	Specifies which subnet groups should be processed in the order of priority.	293
LeaseExpireInterval	LEI	Specifies the interval at which the lease condition of all addresses in the address pool is examined.	294
LeaseTimeDefault	LTD	Specifies the default lease duration for the leases issued by this server.	295
Option	OPTION	Defines options to be passed to the client. Also, within the DefineOptions level, you can define (or redefine) how the option values are handled.	295
PingTime	PT	Specifies a time interval that the server will wait for a response to a ping (ICMP Echo Request).	298
ReservedTime	RT	Specifies the maximum amount of time the server holds an offered address in reserve while waiting for a response from the client.	298
Subnet	SUBNET	Specifies configuration parameters for an address pool administered by a server. An address pool is a range of IP addresses to be leased to clients.	299

Table 17. DHCPD Machine File Statements (continued)

Statement	Minimum Abbreviation	Description	Page
SupportBootP	SB	Specifies whether DHCPD responds to requests from BootP clients.	301
SupportUnlistedClients	SUC	Controls whether DHCPD will respond to clients that are not listed (by client ID) in the machine file.	302
UsedIPAddressExpireInterval	UIPEI	Specifies the amount of time that an IP address, whose lease has expired, will be held before being returned to the available address pool for possible reassignment to another client.	302
Vendor	VENDOR	Specifies an option 43 value to be returned to clients that specify an option 60 value that matches the vendor name.	303

{ (left brace) Statement

Use the left brace statement to indicate the start of a level. Statements which follow the left brace are related to the previous Vendor, Subnet, Class, Client, or DefineOptions statement.

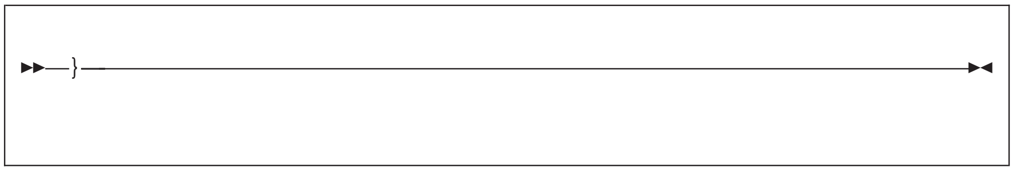
```
►►──{──────────────────────────────────────────────────────────►◄
```

Usage Notes

- The left brace must be the first non-comment statement in the file or the first non-comment statement to follow a Vendor, Subnet, Class, Client or DefineOptions statement.

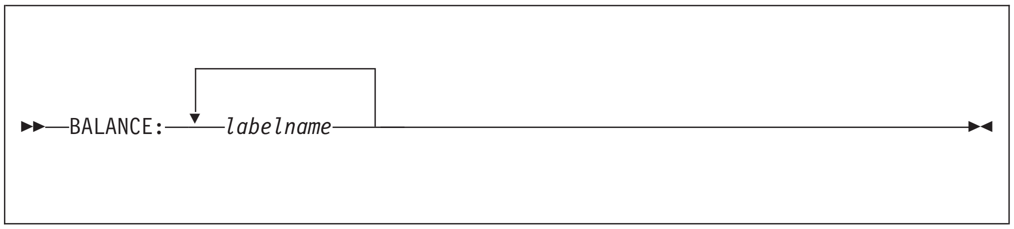
} (right brace) Statement

Use the right brace statement to indicate the end of a level.



Balance: Statement

Use the Balance: statement to specify subnet groups from which addresses should be assigned in an alternating manner.



Operands

labelname
is a label (a string of 1 to 64 alphanumeric characters) that identifies the subnet group.

Usage Notes

- The Balance: statement should be specified at the global level only.
- When the Balance: statement is used, DHCP 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 of Balance: processing of a subnet group. IP addresses are exhausted equally in WIRE1/3 and WIRE1/5: The priority values of "3" and "5" need not be consecutive. There is no other subnet statement for WIRE1 with a priority of 1, 2 or 4.

  ```
  balance: WIRE1
  subnet 9.67.49.0 255.255.255.0 9.67.49.1:9.67.49.100  label:WIRE1/3
  subnet 9.67.48.0 255.255.255.0 9.67.48.1:9.67.48.50  label:WIRE1/5
  ```
- If a label for a subnet group is specified on Subnet statements but not on a Balance: or InOrder: statement, then the label on the subnet has no effect. Clients are allocated addresses for the subnet on which they broadcast the request.

For more information on the format of the Subnet statement, see “SUBNET Statement” on page 299.

BootStrapServer Statement

Use the BootStrapServer statement to specify the IP address of the next server to contact in the boot strap process. Normally, this is the address of the TFTP server from which the client will obtain the boot image after it completes the BootP/DHCP process of acquiring an IP address and basic configuration

```
►►──┬─BOOTSTRAPSERVER─┬──ipaddr────────────────────────────────────►◄
    └─BSS─────────────┘
```

Operands

ipaddr
: is an IP address, specified in dotted-decimal notation, which specifies the address of the next server to contact after completion of the BootP/DHCP protocol step.

Usage Notes

- The BootStrapServer statement may be specified at the global, subnet, class, and client levels only.
- The Boot Strap Server address is returned to the client in the "siaddr" field of BootP and DHCP replies. If the BootStrapServer is not specified in the machine table for the selected response, then the primary host IP address for the host running the DHCPD server is used.

CLASS Statement

Use the Class statement to identify a user-defined group of clients.

```
►►──CLASS──class_name──┬───────┬───────────────────────────────────►◄
                       └─range─┘
```

Operands

class_name
: is the user-defined label that identifies the class. The client specifies the class name using option 77. 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 single (') or double (") quotes.

range
: is 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, 9.17.32.1-9.17.32.128.

  At a global level, a class cannot have a range. A range is only allowed when a class is defined within a subnet. By default, is no range is associated with the class. The range must be a subset of the subnet range.

Usage Notes

- The Class statement should be specified at the global level or subnet level.
- When a Class statement is specified at a Subnet level, the CLASS (and related options) will be used only if:
  - The client specified the class with option 77
  - The client is on the subnet to which the class belongs
  - When an IP address range is specified on the Class statement, then an IP address must be available in the range in order for the Class statement to be used.

  If an IP address has already been indicated on a matching Client statement, that address must be used and must be within the range of addresses listed on the Class statement.
- IP addresses specified on the *range* operand are removed from the general pool of addresses used for the subnet. They will be allocated only to clients that belong to the class. This range can be further restricted by Client statements which specify addresses within the CLASS range.
- A client that requests an IP address from a class which has exhausted its range is offered an IP address from the subnet range, if available. The client is not offered the options associated with the exhausted class.
- To assign configuration parameters such as a lease time for all clients in a class, follow the Class statement with Option statements surrounded by braces.

CLIENT Statement

Use the Client statement to identify a client by an ID, or identify an IP address that should be excluded from a range of available addresses.

```
►►──CLIENT──hw_type──clientID──┬─ipaddr─┬──────────────────────────────►◄
                               ├─ANY────┤
                               └─NONE───┘
```

Operands

hw_type
is the hardware type of the client computer, or 0. The valid client types are defined in STD 2, RFC 1700.

hw_type	**Client hardware**
1	ethernet ether
2	ethernet3 ether3
3	ax.25
4	pronet
5	chaos
6	token-ring tr
7	arcnet
8	hyperchannel
9	lanstar
10	autonet
11	localtalk

12	localnet
13	ultra_link
14	smds
15	frame_relay
16	atm
17	ieee802
18	fddi

clientID
is the hexadecimal MAC address or a name which identifies the client. If a name is specified then:

- *hwtype* must be 0
- If the name contains blanks, then it must be enclosed in single or double quotes.

ipaddr
is an IP address, specified in dotted-decimal notation.

ANY
indicates that the server may choose an address from its pool of available addresses.

NONE
indicates that no response is to be returned to client BootP/DHCP requests.

Usage Notes

- The Client statement may be specified at the global level or subnet level only.
- If the Option statement identifies a client (instead of excluding an address), then it may be followed by left brace ({) and right brace (}) statements, which identify additional statements associated with the Client statement. Option statements are specified within these enclosing braces to associate options with the client. For example:

```
client 6 0000E5E8DC60 any
{
 option 12 ADAM
}
```

- If the Client statement is being used to restrict an IP address from the list of available addresses, both the *hwtype* and *clientid* should be 0. For example:

```
client 0 0 9.100.25.110
```

  While the Client statement can be used to exclude an IP address, a better solution for excluding many IP addresses is to specify a subnet with a range of IP addresses that are never served to clients. For example, the following Subnet statement does not include the first nine addresses on the subnet. These nine addresses are not to be used.

```
subnet 9.100.25.0 255.255.255.0 9.100.25.10-9.100.25.254
```

- The Client statement may be specified at the global level to exclude the client from service by the server, or at a subnet level to exclude a client from service on a particular subnet (address pool). For example:

```
client 6 10005aa4b9ab none
```

- An IP address restricted to a specific client by the *ipaddr* operand is removed from the general pool of addresses used for a subnet. It will be allocated only to the specified client.

DefineOptions Statement

Use the DefineOptions statement to identify a group of options whose processing is being redefined. This allows you to define how the value specified for an option is handled.

```
►►──┬─DEFINEOPTIONS─┬─────────────────────────────────────────────────►◄
    └─DEFOPTS───────┘
```

Usage Notes

- The DefineOptions statement should be specified at the global level only and must precede all Option statements.
- It may be easier to use the DefineOptions statement to define an unrecognized option than coding the value in ASCII or hexadecimal. For example, if a new option (for example 190) is used by the client to contain a series of 32 bit signed time offsets: 18000, 12000, and 3000, you could code it in hex as follows:

  ```
  190 hex 0000465000002ee000000bb8
  ```

  or you could define the options and specify the 190 option values as decimal numbers.

  ```
  DefineOptions
  {
   option 190 lsint(4,*,*)
  }

  option 190  18000 12000 3000
  ```

InOrder: Statement

Use the InOrder: statement to specify subnet groups from which addresses should be assigned based on subnet priorities.

```
               ┌──────────────┐
               ▼              │
►►──INORDER:───┴──labelname───┴───────────────────────────────────────►◄
```

Operands

labelname
  is the labelname associated with a subnet group.

Usage Notes

- The InOrder: statement should be specified at the global level only.
- When the InOrder: statement is used, the subnets that belong to a listed group are processed in the order established by the priority assigned to each subnet. The subnet address pool with the highest priority within a group is completely

exhausted before the subnet address pool with the next highest priority is used. For more information on the Subnet statement, see “SUBNET Statement” on page 299.

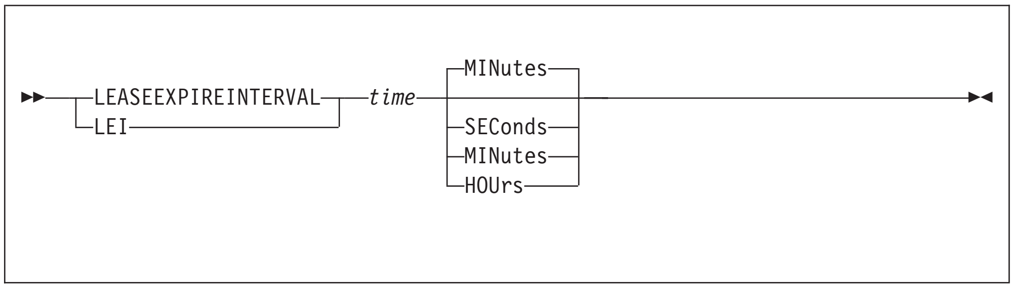
The following is an example of InOrder: processing of subnet group WIRE3. Requests for subnet group WIRE3 exhaust addresses in subnet 9.67.50.0 (WIRE3/1) first, followed by subnet 9.67.51.0 (WIRE3/2), and then 9.67.50.0 (WIRE3/3), which has the same subnet address as WIRE3/1, but specifies a higher address range:

```
inOrder: WIRE3
subnet 9.67.51.0 255.255.255.0 9.67.51.1-9.67.51.50  label:WIRE3/2
subnet 9.67.50.0 255.255.255.0 9.67.50.1-9.67.50.50  label:WIRE3/1
subnet 9.67.50.0 255.255.255.0 9.67.50.51-9.67.50.100  label:WIRE3/3
```

- If a label for a subnet group is specified on Subnet statements but not on a Balance: or InOrder: statement, then the label on the subnet has no effect. Clients are allocated addresses for the subnet on which they broadcast the request.

LeaseExpireInterval Statement

Use the LeaseExpireInterval statement to specify the interval at which the lease condition of all addresses in the address pool is examined.



Operands

time
: is the amount of time to wait between checking all addresses for expired leases.

  The value must be a positive whole number (Base 10) or "-1" (which means infinity). If this statement is not specified, the LeaseExpireInterval defaults to 1 minute. The minimum time value must be equal to at least 15 seconds. The maximum time value must be no more than 12 hours.

SEConds
: indicates the time value is in units of seconds.

MINutes
: indicates the time value is in units of minutes.

HOURs
: indicates the time value is in units of hours.

Usage Notes

- The LeaseExpireInterval statement should be specified at the global level only and therefore, is used for all addresses supported by this server.
- The value specified should be less than the value for LeaseTimeDefault to ensure that expired leases are returned to the pool in a timely manner.

LeaseTimeDefault Statement

Use the LeaseTimeDefault statement to specify the default lease duration for the leases issued by this server.

```
                                        ┌─MINutes──┐
►►──┬─LEASETIMEDEFAULT─┬──time──┼──────────┼────────────────────────────────►◄
    └─LTD──────────────┘        ├─SEConds──┤
                                ├─MINutes──┤
                                ├─HOURs────┤
                                ├─DAYs─────┤
                                ├─MONths───┤
                                └─YEArs────┘
```

Operands

time
: is the default lease duration.

The value must be a positive whole number (Base 10) or "-1" (which means infinity). If this statement is not specified, the LeaseTimeDefault defaults to 24 hours. The minimum time value must be equal to at least 180 seconds.

SEConds
indicates the time value is in units of seconds.

MINutes
indicates the time value is in units of minutes.

HOUrs
indicates the time value is in units of hours.

DAYs
indicates the time value is in units of days.

MONths
indicates the time value is in units of months.

YEArs
indicates the time value is in units of years.

Usage Notes

- The LeaseTimeDefault statement should be specified at the global level only and therefore, is used for all addresses supported by this server.
- The LeaseTimeDefault may be overridden within the machine file using option 51.

OPTION Statement

Use the Option statement to define options to be passed to the client. Also, within the DefineOptions level, you can define (or redefine) option values.

Note: See "DHCP Options" on page 304 for a list of defined option names and option numbers.

```
►►──OPTION──option_number──option_value───────────────────────────────►◄
```

Operands

option_number
is a number from 1 through 254.

option_value
is an option value appropriate for the specified option. When an Option statement is present with a DefineOptions level, the value indicates how the option should be handled when used later in the machine file. In a DefineOptions level, the option values can be:

ASCII
indicates that the option value will be an ASCII string. A provided value can be either a single, blank delimited word, or a string (which may contain blanks), that begins with a single or double quote and ends with the same quote. The character string is translated from EBCDIC to ASCII before its use.

The data may also be specified as a hexadecimal string which is not translated. See the HEX value description that follows for information about how to specify values as hexadecimal strings.

IPADDR
indicates that the allowed option value is an address specified in dotted-decimal notation (e.g. 9.100.25.100)

IPALIST
indicates that the allowed option value is a list of IP addresses in dotted-decimal notation, separated by blanks.

HEX
indicates that the allowed option value is a single blank delimited word that is comprised of EBCDIC character pairs (0-1 and A-F) which represent a hexadecimal byte of data.

UINT(*blen***,***min***,***max***)**
indicates that the allowed option value is an unsigned whole number.

- *blen* is the length, in octets, of the converted value that will be sent to the client.
- *min* is the minimum value allowed for the number. An "*" indicates that the number is the smallest number that can fit in the space allowed for the converted value.
- *max* is the maximum value allowed for the number. An "*" indicates that the number is the largest number that can fit in the space allowed for the converted value.

SINT(*blen***,***min***,***max***)**
indicates that the allowed option value is signed whole number.

- *blen* is the length, in octets, of the converted value that will be sent to the client.

- *min* is the minimum value allowed for the number. An "*" indicates that the number is the smallest number that can fit in the space allowed for the converted value.
- *max* is the maximum value allowed for the number. An "*" indicates that the number is the largest number that can fit in the space allowed for the converted value.

LSINT(*blen***,***min***,***max***)**
indicates that the allowed option value is a list of signed whole numbers, separated by blanks.

- *blen* is the length, in octets, of the converted value for each number that will be sent to the client.
- *min* is the minimum value allowed for a number. An "*" indicates that the number is the smallest number that can fit in the space allowed for the converted value.
- *max* is the maximum value allowed for a number. An "*" indicates that the number is the largest number that can fit in the space allowed for the converted value.

LUINT(*blen***,***min***,***max***)**
indicates that the allowed option value is a list of signed whole numbers, separated by blanks.

- *blen* is the length, in octets, of the converted value for each number that will be sent to the client.
- *min* is the minimum value allowed for a number. An "*" indicates that the number is the smallest number that can fit in the space allowed for the converted value.
- *max* is the maximum value allowed for a number. An "*" indicates that the number is the largest number that can fit in the space allowed for the converted value.

Usage Notes

- The Option statement may be specified at the global, subnet, class, client, vendor or DefineOptions level only.
- The following options should not be specified or redefined within the machine file; they control DHCP processing.

43	Vendor specific information
52	Option overload
53	DHCP message type
54	Server identifier
55	Parameter request list
56	Message
57	Maximum DHCP message size
60	Class identifier
61	Client identifier
77	Client class

- If an option is not defined, then its value is treated as a character string or, if the HEX operand is specified, as hexadecimal data.
- It may be easier to use the DefineOptions statement to define an unrecognized option than coding the value in ASCII or hexadecimal. For example, if a new option (for example 190) is used by the client to contain a series of 32 bit signed time offsets: 18000, 12000, and 3000, you could code it in hex as follows:

```
190 hex 0000465000002ee000000bb8
```

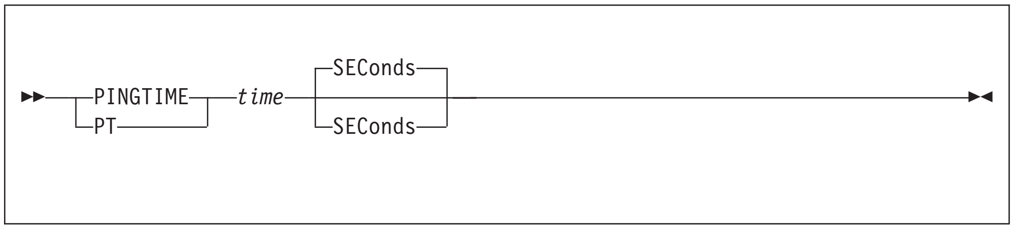
or you could define the options and specify the 190 option values as decimal numbers.

```
DefineOptions
{
 option 190 lsint(4,*,*)
}

option 190  18000 12000 3000
```

- See "DHCP Options" on page 304 for a description of the recognized option numbers and the option values which may be specified.

PingTime Statement

Use the PingTime statement to specify a time interval that the server will wait for a response to a ping (ICMP Echo Request). The server pings an IP address before it is assigned, to make sure that address is not already in use.



Operands

time
: is the number of seconds to wait for the response to a ping. The value must be a whole number (Base 10) between 1 and 30. If this statement is not specified, the PingTime defaults to 1 second.

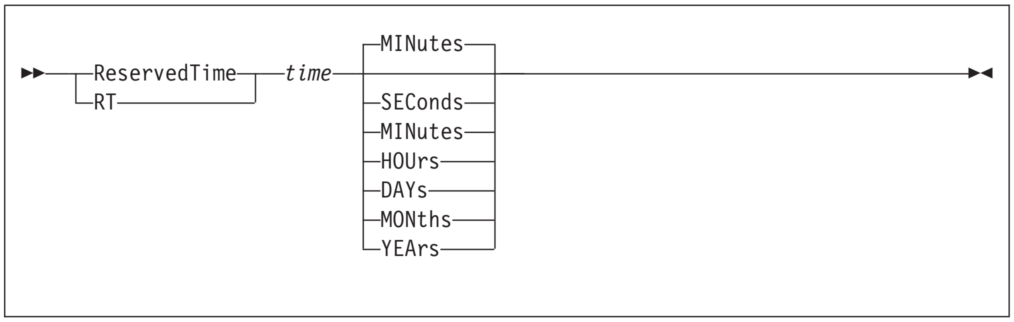
SEConds
: indicates that the time value is seconds.

Usage Notes

- The PingTime statement should be specified at the global level only and therefore, is used for all addresses supported by this server.

ReservedTime Statement

Use the ReservedTime statement to specify the maximum amount of time the server holds an offered address in reserve, while waiting for a response from the client.



Operands

time
: is the amount of time to wait for a response from a client that has been offered an address.

The value must be a positive whole number (Base 10), or "-1" (which means infinity). If this statement is not specified, the ReservedTime defaults to 5 minutes. The minimum time value must be equal to at least 30 seconds.

SEConds
indicates the time value is in units of seconds.

MINutes
indicates the time value is in units of minutes.

HOUrs
indicates the time value is in units of hours.

DAYs
indicates the time value is in units of days.

MONths
indicates the time value is in units of months.

YEArs
indicates the time value is in units of years.

Usage Notes

- The ReservedTime statement should be specified at the global level only and therefore, is used for all addresses issued by this server.

SUBNET Statement

Use the Subnet statement to specify configuration parameters for an address pool administered by a server. An address pool is a range of IP addresses to be leased to clients. The task of configuring subnets also allows you to set lease time and other options for clients that use the address pool. Lease time and other options can be inherited from a global level.

```
►►──SUBNET──subnet_address──┬─────────────┬──range──────────────────────────►
                            └─subnet_mask─┘

►──┬──────────────────────────────┬──┬──────────────┬───────────────────────►◄
   └─label:labelname─┬──────────┬─┘  └─(──comments─┘
                     └─/priority┘
```

Operands

subnet_address
is the address of this subnet, specified in dotted-decimal notation (for example, 9.67.48.0).

subnet_mask
is the mask for the subnet, 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.

A subnet mask can be expressed either in dotted-decimal notation, or as an integer between 8 and 31. For example, the subnet mask 255.255.240.0 is expressed in dotted-decimal notation. For subnet 9.67.48.0, this mask implies an address range from 9.67.48.001 to 9.67.63.254. The equivalent integer format for this mask is 20; the value 20 is the total number of 1s present when the mask is expressed in binary, as 11111111.11111111.11110000.00000000.

Default subnet masks include:

- Class A network - 255.0.0.0
- Class B network - 255.255.0.0
- Class C network - 255.255.255.0

range
is a range of addresses to be administered to this subnet. Enter the 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, 9.67.48.1-9.67.48.128. If more than one subnet has the same subnet address and mask, their ranges should not overlap.

labelname
is a label (a string of 1 to 64 alphanumeric characters) that identifies the subnet group.

priority
is the priority that this subnet is used in relation to other subnets which share the same label.

comments
anything after the left parenthesis is treated as a comment and ignored by the server. This allows compatibility with other IBM DHCP servers.

Usage Notes

- The Subnet statement should be specified at the global level only.
- IP addresses specified on the *range* operand may be restricted from general use in the subnet by CLASS and CLIENT statements that specify IP addresses. The restricted addresses will be allocated only to clients that belong to the related class or client.
- When an address range is specified, do not include the address of the subnet and the address used for broadcast messages. For example, if the subnet address is 9.67.96.0 and the subnet mask is 255.255.240.0, do not include 9.67.96.0 and 9.67.111.255 in the range of addresses.
- Use the Client statement to exclude an IP address from a range of addresses that is administered by the DHCP server. For example, exclude an address that has been permanently assigned to a host. For more information on Client statements, see “CLIENT Statement” on page 291.
- To identify which subnets are grouped together on the same wire, more than one subnet can have the same identifier. For example, the five subnet statements that follow are part of two different groups, labelled as WIRE1 (9.67.48.0 and 9.67.49.0) and WIRE3 (9.67.50.0 and 9.67.51.0). Specifying a label for a subnet group has no affect unless that label is then specified on an InOrder: or Balance: statement. If not specified on one of these two statements, a label is ignored during processing, and clients are allocated addresses for the subnet on which they broadcast their request.

The order in which addresses are selected from subnet statements is determined by the priority associated with each subnet statement. Although second in the sequence of statements for subnet group WIRE3, the address pool for subnet 9.67.50.0 (addresses 1-50) is used before the address pool for subnet 9.67.51.0 — which is defined first in the WIRE3 statement sequence. The address pool for subnet 9.67.50.0 (addresses 51-100) will be used last for the WIRE3 subnet group.

For subnet group WIRE1 (which is listed on the Balance: statement), the assigned priority values have less of an affect on the selection of addresses, because the server will alternate among the members of this group as it attempts to allocate addresses. In this case, the priority values determine only the order in which the list of groups is traversed. Here, an attempt is made to select an address from 9.67.49.0, then 9.67.48.0, then 9.67.49.0, and so on.

```
InOrder: WIRE3
Balance: WIRE1
subnet 9.67.49.0 255.255.255.0 9.67.49.1-9.67.49.100 label:WIRE1/3
subnet 9.67.48.0 255.255.255.0 9.67.48.1-9.67.48.50 label:WIRE1/5
subnet 9.67.51.0 255.255.255.0 9.67.51.1-9.67.51.50  label:WIRE3/2
subnet 9.67.50.0 255.255.255.0 9.67.50.1-9.67.50.50  label:WIRE3/1
subnet 9.67.50.0 255.255.255.0 9.67.50.51-9.67.50.100  label:WIRE3/3
```

SupportBootP Statement

Use the SupportBootP statement to specify whether DHCPD responds to requests from BootP clients.

```
►►──┬─SUPPORTBOOTP─┬──┬─YES─┬──────────────────────────────────────────────►◄
    └─SB───────────┘  └─NO──┘
```

Operands

YES
indicates that BootP clients should be supported. This is the default if the statement is not specified.

NO
indicates that BootP clients should not be supported.

Usage Notes

- The SupportBootP statement should be specified at the global level only and therefore, is used for all BootP requests received by this server.
- If DHCPD previously supported BootP clients and has been reconfigured to not support BootP clients, the address binding for any BootP clients that was established before the reconfiguration will be maintained until the BootP client sends another request (when it is restarting). At that time, the server will not respond, and the binding will be removed.

SupportUnlistedClients Statement

Use the SupportUnlistedClients statement to control whether DHCPD will respond to clients that are not listed (by client ID) in the machine file.

```
►►──┬─SUPPORTUNLISTEDCLIENTS─┬──┬─YES─┬──────────────────────────────►◄
    └─SUC────────────────────┘  └─NO──┘
```

Operands

YES
: indicates that clients not listed (by client ID) in the machine file should be supported.

NO
: indicates that only clients listed (by client ID) in the machine file should be supported.

Usage Notes

- The SupportUnlistedClients statement should be specified at the global level only and therefore, is used for all requests received by this server.
- This statement can be used to limit access to addresses managed by this DHCP server. Listing the client IDs for all acceptable clients may be time consuming.
- If the SupportUnlistedClients statement is not specified, the default is ’YES’ — unlisted clients are supported.

UsedIPAddressExpireInterval Statement

Use the UsedIPAddressExpireInterval statement to specify the amount of time to retain an IP address whose lease has expired, before that address is returned to the available address pool for possible reassignment to another client. This allows the client that last used an address to again request it — within the specified time interval — and have that address reassigned to it.

```
                                             ┌─MINutes──┐
►►──┬─USEDIPADDRESSEXPIREINTERVAL─┬──time──┤          ├──────────►◄
    └─UIPEI───────────────────────┘        ├─SEConds──┤
                                             ├─MINutes──┤
                                             ├─HOUrs────┤
                                             ├─DAYs─────┤
                                             ├─MONths───┤
                                             └─YEArs────┘
```

Operands

time
: is the amount of time to wait before an IP address, whose lease has expired, is returned to the pool of available addresses.

The value must be a positive whole number (Base 10), or "-1" (which means infinity). If this statement is omitted, the UsedIPAddressExpireInterval defaults to 30 seconds.

SEConds
indicates the time value is in units of seconds.

MINutes
indicates the time value is in units of minutes.

HOUrs
indicates the time value is in units of hours.

DAYs
indicates the time value is in units of days.

MONths
indicates the time value is in units of months.

YEArs
indicates the time value is in units of years.

Usage Notes

- The UsedIPAddressExpireInterval statement should be specified at the global level only and therefore, is used for all addresses supported by this server.

VENDOR Statement

Use the Vendor statement to specify an option 43 value to be returned to clients that specify an option 60 value that matches the vendor name.

```
►►──VENDOR──vendor_name──┬──────────────────┬──────────────────────────────►◄
                         └─HEX──hexvalue────┘
```

Operands

vendor_name
is the user-defined label that identifies the vendor. The client transmits this label using option 60. 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 single (') or double quotes (").

HEX
indicates that option 43 data follows this keyword.

hexvalue
is an EBCDIC representation of the hexadecimal value to be returned to the client in option 43.

Usage Notes

- The Vendor statement should be specified at the global level only.
- The Vendor statement may be followed by a pair of braces that delimit the options particular to this vendor. Within these braces, the usual option value encoding and decoding rules do not apply. An option 43 value will be

constructed from the specified options in the form of an option number, followed by the length of the data for that option, and the option data.

The value for each option must be specified either as an EBCDIC string, or in hexadecimal form, by specifying the HEX keyword followed by the EBCDIC representation of the hexadecimal value.

For example

```
VENDOR IBM
{
 OPTION 1 1
 OPTION 15 HEX F1
}
```

would create an option 43 value of (shown in hex):

```
0101300F01F1
```

Note: The value of 1 for option 1 is ’30’ which is the ASCII value for the character ’1’.

You could also specify the value on the Vendor statement using the hex keyword, as follows:

```
VENDOR IBM hex 0101300F01F1
```

DHCP Options

The following table lists the option names, their numbers, and the locations of additional information.

Table 18. DHCP Option Names and Numbers

Name	Number	Page
All Subnets are local	27	311
ARP Cache Timeout	35	313
Boot File name	67	318
Boot File Size	13	309
Broadcast Address	28	312
Client Class	77	320
Client Identifier	61	317
Cookie Server	8	308
Default Finder Server	73	319
Default Internet Relay Chat (IRC) Server	74	319
Default IP Time-to-live	23	311
Default World Wide Web (WWW) Server	72	319
DHCP Message Type	53	316
Domain Name	15	309
Domain Name Server	6	308
Ethernet Encapsulation	36	313
Extension Path	18	310
Host Name	12	309
Impress Server	10	308
Interface MTU	26	311

Table 18. DHCP Option Names and Numbers (continued)

Name	Number	Page
IP Address Lease Time	51	316
IP Forwarding Enable/Disable	19	310
Log Server	7	308
LPR Server	9	308
Mask Supplier	30	312
Maximum Datagram Reassembly Size	22	310
Maximum DHCP Message Size	57	317
Merit Dump File	14	309
Message	56	317
Mobile IP Home Agent	68	318
Name Server	5	307
NetBIOS over TCP/IP Datagram Distribution Server	45	315
NetBIOS over TCP/IP Name Server	44	315
NetBIOS over TCP/IP Node Type	46	315
NetBIOS over TCP/IP Scope	47	315
Network Information Servers	41	314
Network Information Service Domain	40	314
Network Information Service+ Domain	64	317
Network Information Service+ Servers	65	318
Network News Transport Protocol (NNTP) Server	71	319
Network Time Protocol Servers	42	314
Non-Local Source Routing Enable/Disable	20	310
Option Overload	52	316
Parameter Request List	55	316
Path MTU Aging Timeout	24	311
Path MTU Plateau Table	25	311
Perform Mask Discovery	29	312
Perform Router Discovery	31	312
Policy Filter	21	310
Post Office Protocol (POP3) Server	70	319
Rebinding (T2) Time Value	59	317
Renewal (T1) Time Value	58	317
Requested IP address	50	315
Resource Location Server	11	308
Root Path	17	309
Router Address	3	307
Router Solicitation Address	32	312
Server Identifier	54	316
Simple Mail Transport Protocol (SMTP) Server	69	318
Static Route	33	313

Table 18. DHCP Option Names and Numbers (continued)

Name	Number	Page
StreetTalk Directory Assistance (STDA) Server	76	320
StreetTalk Server	75	319
Subnet Mask	1	307
Swap Server	16	309
TCP Default TTL	37	313
TCP Keepalive Interval	38	314
TCP Keepalive Garbage	39	314
TFTP Server Name	66	318
Time Offset	2	307
Time Server	4	307
Trailer Encapsulation	34	313
Vendor Class Identifier	60	317
Vendor Specific Information	43	314
X Window System Display Manager	49	315
X Window System Font Server	48	315

Table 20 on page 307 lists the DHCP option numbers, their description and the type of data which may be entered in the machine file for the option on the Option statement.

The following table lists the complex datatypes that appear in Table 20 on page 307.

Table 19. DHCP Option Datatype Formats

Datatype	Format
Character string	A single blank delimited word, or a string (which may contain blanks), that begins with a single or double quote and ends with the same quote. The character string is translated from EBCDIC to ASCII before its use.
Hexadecimal string	"HEX" followed by a single blank delimited word containing pairs of EBCDIC characters (0-1, and A-F) which represent a hexadecimal byte of data. The data is assumed to represent ASCII or OCTET values and is not translated from EBCDIC to ASCII as character strings are translated.
IP Address	Dotted-decimal notation (for example, 9.100.25.100)
IP Address List	List of IP addresses separated by blanks
IP Address Pair	Two IP addresses in dotted-decimal notation which are separated by blanks.
IP Address Pair List	List of IP address pairs. All addresses are separated from each other by blanks.
Signed Integer	A single blank delimited word that begins with a sign (+ or –). If a sign is not specified, "+" is the default.
Unsigned Integer	A single blank delimited word that does not begin with a sign and denotes a positive value.
List of Signed Integers	A list of blank delimited numbers that begin with a sign (+ or –). If a sign is not specified, "+" is the default.

Table 19. DHCP Option Datatype Formats (continued)

Datatype	Format
List of Unsigned Integers	A list of blank delimited numbers that do not begin with a sign and denote a positive value.

Table 20. DHCP Options

Option Number	Option Information
1	**Name:** SUBNET MASK
	Description: Specifies the client's subnet mask per RFC 950.
	Datatype: IP address
2	**Name:** Time Offset
	Description: Specifies the offset of the client's subnet in seconds from Coordinated Universal Time (UTC).
	Datatype: Signed Integer **Minimum** -2,147,483,647 **Maximum** 2,147,483,647
	Notes: 1. A positive offset indicates a location east of the zero meridian and a negative offset indicates a location west of the zero meridian.
3	**Name:** Router Addresses
	Description: Specifies a list of IP address for routers on the client's subnet.
	Datatype: IP Address List
	Notes: 1. Routers should be listed in order of preference.
4	**Name:** Time Server
	Description: Specifies a list of RFC 868 time servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
5	**Name:** Name Server
	Description: Specifies a list of IEN 116 name servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.

Table 20. DHCP Options (continued)

Option Number	Option Information
6	**Name:** Domain Name Server
	Description: Specifies a list of Domain Name System (STD 13, RFC 1035) name servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
7	**Name:** Log Server
	Description: Specifies a list of MIT-LCS UDP log servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
8	**Name:** Cookie Server
	Description: Specifies a list of RFC 865 cookie servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
9	**Name:** LPR Server
	Description: Specifies a list of RFC 1179 line printer servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
10	**Name:** Impress Server
	Description: Specifies a list of IMAGEN Impress servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
11	**Name:** Resource Location Server
	Description: Specifies a list of RFC 886 Resource Location servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.

Table 20. DHCP Options (continued)

Option Number	Option Information
12	**Name:** Host Name
	Description: Specifies the name of the client.
	Datatype: Character string or hexadecimal string
	Notes: 1. The name may or may not be qualified with the local domain. If the Domain Name option is specified then this option must not include the local domain name as part of the character string. 2. See RFC 1035 for character set restrictions.
13	**Name:** Boot File Size
	Description: Specifies the length (in 512 octet blocks) of the default boot image for the client.
	Datatype: Unsigned Integer **Maximum** 65,535
14	**Name:** Merit Dump File
	Description: Specifies the path name of a file to which the client's core image should be dumped in the event the client crashes.
	Datatype: Character string or hexadecimal string
15	**Name:** Domain Name
	Description: Specifies the domain name that the client should use when resolving host names via the Domain Name System.
	Datatype: Character string or hexadecimal string
	Notes: 1. See the Host Name option for information on specifying these options together.
16	**Name:** Swap Server
	Description: Specifies the IP address of the client's swap server.
	Datatype: IP Address
17	**Name:** Root Path
	Description: Specifies the path name that contains the client's root disk.
	Datatype: Character string or hexadecimal string

Table 20. DHCP Options (continued)

Option Number	Option Information
18	**Name:** Extensions Path
	Description: Specifies a file, retrievable using TFTP, that contains information which can be interpreted in the same way as the 64 octet vendor-extension field within the BOOTP response.
	Datatype: Character string or hexadecimal string
	Notes: 1. The file length of the specified file is constrained only by the host file system that contains the file. 2. All references to option 18 (this option) within the file are ignored by the client.
19	**Name:** IP Forwarding Enable/Disable
	Description: Specifies whether the client should configure its IP layer for packet forwarding. A value of 0 means disable IP forwarding, and a value of 1 means enable IP forwarding.
	Datatype: Integer, 0, or 1
20	**Name:** Non-Local Source Routine Enable/Disable
	Description: Specifies whether a client should configure its IP layer to allow forwarding of datagrams with non-local source routes. A value of 0 means that forwarding of such datagrams are not allowed, and a value of 1 means that forwarding is allowed.
	Datatype: Integer, 0, or 1
21	**Name:** Policy Filter
	Description: Specifies policy filters for non-local source routing. The filters consist of a list of IP addresses and masks which specify destination/ mask pairs with which to filter incoming source routes. Any source routed datagram whose next hop address does not match one of the filters will be discarded by the client.
	Datatype: IP Address Pair List
22	**Name:** Maximum Datagram Reassembly Size
	Description: Specifies the maximum size datagram that the client should be prepared to reassemble.
	Datatype: Unsigned Integer **Minimum** 576 **Maximum** 65,535

Table 20. DHCP Options (continued)

Option Number	Option Information
23	**Name:** Default IP Time-to-live
	Description: Specifies the default time-to-live that the client should use on outgoing datagrams.
	Datatype: Unsigned integer **Minimum** 1 **Maximum** 255
24	**Name:** Path MTU Aging Timeout
	Description: Specifies the timeout (in seconds) to use, when aging Path MTU values discovered by the mechanism defined in RFC 1191.
	Datatype: Unsigned integer **Maximum** 4,294,967,295
25	**Name:** Path MTU Plateau Table
	Description: Specifies a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is formatted as a list of 16-bit unsigned integers, ordered from smallest to largest.
	Datatype: List of unsigned integers **Minimum** 68 **Maximum** 65,535
26	**Name:** Interface MTU
	Description: Specifies the MTU to use on this interface.
	Datatype: Unsigned integer **Minimum** 68 **Maximum** 65,535
27	**Name:** All Subnets are Local
	Description: Specifies whether the client may assume that all subnets of the IP network use the same MTU as the subnet of that network to which the client is directly connected. A value of 1 means that all subnets share the same MTU. A value of 0 means that the client should assume that some subnets of the directly connected network may have smaller MTUs.
	Datatype: Integer, 0, or 1

Table 20. DHCP Options (continued)

Option Number	Option Information
28	**Name:** Broadcast Address
	Description: Specifies the broadcast address in use on the client's subnet.
	Datatype: IP Address
	Notes: Legal values for broadcast addresses are specified in STD 3, RFC 1122.
29	**Name:** Perform Mask Discovery
	Description: Specifies whether the client should perform subnet mask discovery using ICMP. A value of 0 indicates that the client should not perform mask discovery. A value of 1 means that the client should perform mask discovery.
	Datatype: Integer, 0, or 1
30	**Name:** Mask Supplier
	Description: Specifies whether the client should respond to subnet mask requests using ICMP. A value of 0 indicates that the client should not respond. A value of 1 means that the client should respond.
	Datatype: Integer, 0, or 1
31	**Name:** Perform Router Discovery
	Description: Specifies whether the client should solicit routers using the Router Discover mechanism defined in RFC 1256. A value of 0 indicates that the client should not perform router discovery. A value of 1 means that the client should perform router discovery.
	Datatype: Integer, 0, or 1
32	**Name:** Router Solicitation Address
	Description: Specifies the IP address to which the client should transmit router solicitation requests.
	Datatype: IP Address

Table 20. DHCP Options (continued)

Option Number	Option Information
33	**Name:** Static Route
	Description: Specifies a list of static routes that the client should install in its routing cache. The routes consist of pairs of IP addresses. The first address is the destination address, and the second address is the router for the destination.
	Datatype: IP Address Pair List
	Notes: 1. If multiple routes to the same destination are specified, they should be listed in descending order of priority. 2. The default route (0.0.0.0) is an illegal destination for a static route.
34	**Name:** Trailer Encapsulation
	Description: Specifies whether the client should negotiate the use of trailers (RFC 893) when using the ARP protocol. A value of 0 indicates that the client should not attempt to use trailers. A value of 1 means that the client should attempt to use trailers.
	Datatype: Integer, 0, or 1
35	**Name:** ARP Cache Timeout
	Description: Specifies the timeout in seconds for ARP cache entries.
	Datatype: Unsigned Integer **Maximum** 4,294,967,295
36	**Name:** Ethernet Encapsulation
	Description: Specifies whether the client should use Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation if the interface is an Ethernet. A value of 0 indicates that the client should use RFC 894 encapsulation. A value of 1 indicates that the client should use RFC 1042 encapsulation.
	Datatype: Integer, 0, or 1
37	**Name:** TCP Default TTL
	Description: Specifies the default TTL that the client should use when sending TCP segments.
	Datatype: Unsigned integer **Maximum** 255

Table 20. DHCP Options (continued)

Option Number	Option Information
38	**Name:** TCP Keepalive Interval
	Description: Specifies the interval (in seconds) that a TCP client should wait before sending a keepalive message on a TCP connection. A value of zero indicates that the client should not generate keepalive messages on connections unless specifically requested to do so by an application.
	Datatype: Unsigned Integer **Maximum** 4,294,967,295
39	**Name:** TCP Keepalive Garbage
	Description: Specifies whether the client should send TCP keepalive messages with an octet of garbage for compatibility with older implementations. A value of 0 indicates that a garbage octet should not be sent. A value of 1 indicates that a garbage octet should be sent.
	Datatype: Integer, 0, or 1
40	**Name:** Network Information Service Domain
	Description: Specifies the name of the client's NIS domain.
	Datatype: Character string or hexadecimal string
41	**Name:** Network Information Servers
	Description: Specifies a list of IP addresses indicating NIS servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
42	**Name:** Network Time Protocol Servers
	Description: Specifies a list of IP addresses indicating NTP servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
43	**Name:** Vendor Specific Information
	Description: Specifies hexadecimal data to return to the client. This information is vendor specific.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the Vendor statement only.

Table 20. DHCP Options (continued)

Option Number	Option Information
44	**Name:** NetBIOS over TCP/IP Name Server
	Description: Specifies a list of RFC 1001/1002 NBNS name servers.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
45	**Name:** NetBIOS over TCP/IP Datagram Distribution Server
	Description: Specifies a list of RFC 1001/1002 NBDD servers.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
46	**Name:** NetBIOS over TCP/IP Node Type
	Description: Specifies the client type.
	Datatype: Integer, 1, 2, 4 or 8
47	**Name:** NetBIOS over TCP/IP Scope
	Description: Specifies the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002.
	Datatype: Character string or hexadecimal string
48	**Name:** X Window System Font Server
	Description: Specifies a list of X Window System Font Servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
49	**Name:** X Window System Display Manager
	Description: Specifies a list of IP address of systems that are running the X Window System Display Manager and are available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
50	**Name:** Network Information Service+ Domain
	Description: Specifies the name of the client’s NIS+ domain.
	Datatype: Character string or hexadecimal string

Table 20. DHCP Options (continued)

Option Number	Option Information
51	**Name:** IP Address Lease Time
	Description: Specifies maximum default lease time in seconds. This option is used in a server reply (DHCPOFFER) to specify the lease time the server is willing to offer.
	Datatype: Unsigned integer **Maximum** 4,294,967,295
	Notes: 1. This option overrides the lease time specified by the DHCPD LEASETIMEDEFAULT statement.
52	**Name:** Option Overload
	Description: Specifies which fields are being overloaded to carry DHCP options. A DHCP server inserts this option if the returned parameters will exceed the usual space allowed for options. **1** means that the ’file’ field is used to hold options. **2** means that the ’sname’ field is used to hold options. **3** means that both the ’file’ and ’sname’ field are used to hold options.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the server to control processing.
53	**Name:** DHCP Message Type
	Description: Specifies the type of DHCP message being sent.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the server and the client to control processing.
54	**Name:** Server Identifier
	Description: Specifies the IP address of the DHCP server. DHCP Servers use this option to distinguish their DHCPOFFER responses from other servers’ responses. DHCP clients use the contents of this option as the destination address for any DHCP message unicast to the server. They also use this option to indicate which server’s offer is being accepted.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the server and the client to control processing.
55	**Name:** Parameter Request List
	Description: Specifies a list of options requested by the client and the order which the client would like them to appear.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the server and the client to control processing.

Table 20. DHCP Options (continued)

Option Number	Option Information
56	**Name:** Message
	Description: Specifies an error message to be passed between the client and the server.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the server and the client to control processing.
57	**Name:** Maximum DHCP Message Size
	Description: Specifies the maximum length DHCP message that the client is willing to receive.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the client to control processing.
58	**Name:** Renewal (T1) Time Value
	Description: Specifies the time interval (in seconds) from address assignment until the client transitions to the renewing state.
	Datatype: Unsigned Integer **Maximum** 4,294,967,295
59	**Name:** Rebinding (T2) Time Value
	Description: Specifies the time interval (in seconds) from address assignment until the client transitions to the rebinding state.
	Datatype: Unsigned Integer **Maximum** 4,294,967,295
60	**Name:** Vendor Class
	Description: Specifies a string which identifies the class of vendor. This information is compared to the value specified on the Vendor machine statement as the vendor name.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the client to control processing.
61	**Name:** Client Identifier
	Description: Specifies a unique identifier for the client.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the client to control processing.
64	**Name:** Network Information Service+ Domain
	Description: Specifies the name of the client's NIS+ domain.
	Datatype: Character string or hexadecimal string

Table 20. DHCP Options (continued)

Option Number	Option Information
65	**Name:** Network Information Service+ Servers
	Description: Specifies a list of IP addresses indicating NIS+ servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be specified in order of preference.
66	**Name:** TFTP Server Name
	Description: Specifies the name of the TFTP server which the client should use.
	Datatype: Character string or hexadecimal string
	Notes: 1. This value is passed to the client in the 'sname' field or as an option, if option overloading occurs. See option 52 for information on overloading.
67	**Name:** Bootfile name
	Description: Specifies the name of the bootfile for the client to request.
	Datatype: Character string or hexadecimal string
	Notes: 1. This value is passed to the client in the 'file' field or as an option, if option overloading occurs. See option 52 for information on overloading.
68	**Name:** Mobile IP Home Agent
	Description: Specifies a list of IP addresses indicating mobile IP home agents available to the client.
	Datatype: IP Address List
	Notes: 1. Agents should be listed in order of preference. 2. A 0 may be specified instead of an IP address list to indicate that no home agents are available.
69	**Name:** Simple Mail Transport Protocol (SMTP) Server
	Description: Specifies a list of IP addresses indicating SMTP servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.

Table 20. DHCP Options (continued)

Option Number	Option Information
70	**Name:** Post Office Protocol (POP3) Server
	Description: Specifies a list of IP addresses indicating POP3 servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
71	**Name:** Network News Transport Protocol (NNTP) Server
	Description: Specifies a list of IP addresses indicating NNTP servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
72	**Name:** Default World Wide Web (WWW) Server
	Description: Specifies a list of IP addresses indicating WWW servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
73	**Name:** Default Finger Server
	Description: Specifies a list of IP addresses indicating Finger servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
74	**Name:** Default Internet Relay Chat (IRC) Server
	Description: Specifies a list of IP addresses indicating IRC servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
75	**Name:** StreetTalk Server
	Description: Specifies a list of IP addresses indicating StreetTalk servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.

Table 20. DHCP Options (continued)

Option Number	Option Information
76	**Name:** StreetTalk Directory Assistance (STDA) Server
	Description: Specifies a list of IP addresses indicating STDA servers available to the client.
	Datatype: IP Address List
	Notes: 1. Servers should be listed in order of preference.
77	**Name:** Client Class
	Description: Specifies a character string which identifies the client's class. Option 77 is compared to the *class_name* value specified on the machine file Class statement.
	Notes: **RESTRICTED.** This option may not be specified. It is generated by the client to control processing.

| ## Dynamic Server Operation

| Some configuration attributes (such as tracing and client request handling) can be
| changed during server execution using the DHCPD subcommands described in the
| next section. In addition, certain server activities can be queried or changed by
| subcommands. Any subcommand not understood by DHCP is assumed to be a
| CMS command and is passed to the CMS command line for execution.

| Issue DHCPD subcommands at the DHCP server console.

DHCPD Subcommands

The DHCPD subcommands are listed in Table 21. This table provides the shortest abbreviation, a description, and a page reference for more information for each DHCPD subcommand.

Table 21. DHCPD Subcommands

Subcommand	Minimum Abbreviation	Description	Page
CMS	CMS	Passes a command to CMS for execution.	321
CONFIG	CONFIG	Displays configuration information.	322
DELETE	DELETE	Deletes an IP address lease that has been allocated to a client.	324
EXCLUDE	EXCLUDE	Identifies adapter addresses for which BootP/DHCP requests should be ignored.	325
EXIT	EXIT	Stops the DHCP server and its processing. EXIT is equivalent to QUIT and STOP.	326
FORWARD	FORWARD	Controls the forwarding of BootP/DHCP requests to another DHCP server.	326
HELP	HELP	Displays a summary of DHCPD subcommands.	328
INCLUDE	INCLUDE	Identifies adapter addresses for which BootP/DHCP requests should be handled.	328

Table 21. DHCPD Subcommands (continued)

Subcommand	Minimum Abbreviation	Description	Page
LURK	LURK	Toggles the LURK mode of the DHCP server.	329
QUIT	QUIT	Stops the DHCP server and its processing. QUIT is equivalent to EXIT and STOP.	329
RELOAD	RELOAD	Reloads DHCPD machine and configuration files.	329
SHOW	SHOW	Displays the information which would be returned to a client that specified parameters similar to the ones specified on the subcommand.	331
STATUS	STATUS	Displays the status of an IP address, Client or Subnet's addresses.	334
STAYUP	STAYUP	Toggles the STAYUP mode of the DHCP server.	336
STOP	STOP	Stops the DHCP server and its processing. STOP is equivalent to EXIT and QUIT.	336
TRACE	TRACE	Toggles the TRACE mode of the DHCP server.	336

CMS Subcommand

Use the CMS subcommand to issue a command to CMS.

```
►►──┬─────┬──cms_command──────────────────────────────────────►◄
    └─CMS─┘
```

Operands

cms_command
   is the CMS command to be issued.

Usage Notes

- Do not issue any CMS command that would take considerable time to execute (for example, XEDIT). While the CMS command executes, the server does not respond to requests.
- The CMS keyword is usually not required because the daemon will pass any command string that is not recognized as a DHCPD subcommand to CMS. The CMS keyword is used to identify CMS commands which would normally be interpreted as a DHCPD subcommand, for example, TRACE.

Examples

- After completion of any command, the following ready prompt is displayed:

  ```
  DHCPD Ready;
  ```

  or

  ```
  DHCPD Ready (rc);
  ```

  if the return code (rc) is not zero.

CONFIG Subcommand

Use the CONFIG subcommand to display configuration information.

```
►►──CONFIG────────────────────────────────────────────────────────────────►◄
```

Usage Notes

- The CONFIG subcommand causes the DHCP daemon to query the current TCP/IP configuration of the host where the DHCP daemon is running, to determine the IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand’s operation, whether the subcommand completes successfully or not.

Examples

- The CONFIG subcommand produces a multiple line response which indicates the status of settings, table files used, included and excluded adapter addresses, and forwards that are active or could be activated. This output is discussed below, in sections, for clarity of meaning.

```
Lurk=l Stayup=s Trace=t
Machine Table=filespec
Configuration Table=filespec
```

This section indicates the status of the LURK, STAYUP, and TRACE settings, along with the file specifications for the machine and configuration table files used by the server.

l is 1 if LURK mode is on; 0 if LURK mode is off.

s is 1 if STAYUP mode is on; 0 if STAYUP mode is off.

t is 1 if TRACE mode is on; 0 if TRACE mode is off.

filespec
is the file name, file type and file mode of the file that is in use.

```
Included Addresses:
  Adapter adpaddr reqtype
```

This section indicates the IP addresses of adapters for which the DHCP daemon will process requests. One "Adapter" line is displayed for each adapter that the DHCP daemon will handle. If there are no included addresses, "NONE" is displayed instead of the "Adapter" line(s).

adpaddr
is the IP address of an adapter.

reqtype
is the type of request that will be handled. Possible values are:

CLIENTS	for BootP/DHCP requests broadcast by clients to the host.
GATEWAYS	for BootP/DHCP requests forwarded by a DHCP daemon on behalf of a client.
ANY	for any client or gateway forwarded requests.

```
Excluded Addresses:
  Adapter adpaddr reqtype
```

This section indicates the IP addresses of adapters the DHCP daemon should ignore. If there are no excluded addresses, "NONE" is displayed instead of the "Adapter" line(s).

adpaddr
is the IP address of the adapter.

reqtype
is the type of request that will be excluded. Possible values are:

CLIENTS	for BootP/DHCP requests broadcast by clients to the host.
GATEWAYS	for BootP/DHCP requests forwarded by a DHCP daemon on behalf of a client.
ANY	for any client or gateway forwarded requests.

```
Forwards:
  Adapter adpaddr -> toaddr frequency actflag
  Gateway gateaddr -> toaddr frequency
```

This section indicates whether BootP/DHCP request forwarding has been specified for:
– requests received on specific adapters, or
– requests forwarded by a specific gateway.

adpaddr
is the IP address of the adapter.

gateaddr
is a gateway IP address. The gateway IP address is the address of an adapter on which a request is initially received, but is then forwarded.

toaddr
is the IP address of a host that is running another DHCP daemon which should receive forwarded requests.

frequency
is an indication of when forwarding should occur for the specified adapter or gateway. ALWAYS indicates that any request on the adapter or forwarded by the gateway should always be forwarded.

actflag
indicates the status of the adapter for which the forward has been specified. This value can be either:

INCLUDED	indicating that the adapter is included in the configuration and handles both client and gateway-forwarded requests.
EXCLUDED	indicating that the adapter is excluded from the configuration (for example, no forwarding will occur until the adapter is included in the configuration).
PARTIAL	indicating that some BootP/DHCP requests received on the adapter will not be handled. This can occur if the EXCLUDE or INCLUDE subcommand resulted in some BootP/DHCP requests not being handled. For example, gateway forwarded requests received over a specific adapter may be

excluded, while client requests may be included. For more information about how to control request handling see “INCLUDE Subcommand” on page 265 and “EXCLUDE Subcommand” on page 261.

DELETE Subcommand

Use the DELETE subcommand to remove an active lease for an IP address that has been given to a client.

```
►►──DELETE──LEASE──┬─ADDress──ipaddr──────────┬────────────────────────────────►◄
                   └─CLIent──hwtype──clientid─┘
```

Operands

LEASE
: indicates that a lease is being deleted.

ADDress
: indicates that the operand that follows is the IP address whose lease is to be deleted.

ipaddr
: is an IP address specified in dotted-decimal notation.

CLIent
: indicates that the operands that follow identify the client that is associated with the IP address whose lease will be deleted.

hwtype
: is the hardware type of the client computer, or 0. The valid client types are defined in STD 2, RFC 1700.

hwtype	Client hardware
1	ethernet ether
2	ethernet3 ether3
3	ax.25
4	pronet
5	chaos
6	token-ring tr
7	arcnet
8	hyperchannel
9	lanstar
10	autonet
11	localtalk
12	localnet
13	ultra_link
14	smds
15	frame_relay
16	atm
17	ieee802
18	fddi

clientID
: is the hexadecimal MAC address or a name which identifies the client. If a name is specified then:

- *hwtype* must be 0
- If the name contains blanks, then it must be enclosed in single or double quotes.

Usage Notes

- The DELETE subcommand is useful when you determine that an assigned lease is no longer being used and you wish to make the address available for reassignment. For example, a lease will become available when a BootP client, which has a permanent lease on an address, moves to a different subnet.

EXCLUDE Subcommand

Use the EXCLUDE subcommand to specify an adapter address to ignore. You can specify additional operands to indicate whether all requests received across a specific adapter should be ignored, or whether only client BootP/DHCP requests or gateway-forwarded requests should be ignored.

```
                                    ┌─ANY──────┐
►►──EXCLUDE──┬─ADApter──ipaddr─┬──┼──────────┼──────────────────────────►◄
             └─ALL─────────────┘  ├─CLIents──┤
                                    ├─GATeways─┤
                                    └─ANY──────┘
```

Operands

ADApter *ipaddr*
indicates the IP address of the adapter on the host system that should be ignored.

ALL
indicates that all adapters should be ignored.

ANY
indicates that any request that is received on the specified adapters (or "ALL") should be ignored. This is the default.

CLIents
indicates that only client BootP/DHCP requests that are received on the specified adapters (or "ALL") should be ignored.

GATeways
indicates that gateway-forwarded requests that are received on the specified adapters (or "ALL") should be ignored.

Usage Notes

- The opposite of the EXCLUDE subcommand is the INCLUDE subcommand. Any values set by the EXCLUDE subcommand may be reset by the INCLUDE subcommand.
- The EXCLUDE subcommand causes the DHCP daemon to query the current TCP/IP configuration of the host where the DHCP daemon is running, to determine the IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand's operation, whether the subcommand completes successfully or not.

EXIT Subcommand

Use the EXIT subcommand to stop the DHCP daemon. This subcommand is equivalent to the QUIT and STOP subcommands.

```
►►──EXIT──────────────────────────────────────────────────────────────►◄
```

Operands

The EXIT subcommand has no operands.

FORWARD Subcommand

Use the FORWARD subcommand to specify BootP/DHCP requests that should be forwarded to another DHCP daemon at another IP address. Requests are selected on the basis of the adapter on which they are received or the gateway from which they were forwarded.

```
                                                      ┌─ALWAYS──┐
►►──FORWARD──┬─ADApter──ipaddr───┬──┬──────┬──toipaddr──┼─────────┼──────►◄
             ├─GATeway──gateaddr─┤  └─TO───┘            ├─ALWays──┤
             └─ALL───────────────┘                      └─NEVer───┘
```

Operands

ADApter *ipaddr*
: indicates that BootP/DHCP requests received over the specified IP address should be forwarded. The IP address is the address of an adapter on the host system that would receive the request.

GATeway *gateaddr*
: indicates that BootP/DHCP requests that were forwarded by a gateway at the specified IP address should be forwarded.

ALL
: indicates that all IP adapter addresses should be handled as forwarding addresses.

TO *toipaddr*
: indicates the IP address to which the BootP/DHCP request should be forwarded.

ALWAYS
: indicates that BootP/DHCP requests that pass the selection criteria (for example, on the specified adapter or from the specified gateway) should always be forwarded. This is the default.

NEVER
: indicates that BootP/DHCP requests that pass the selection criteria (for example, on the specified adapter or from the specified gateway) should never be forwarded. This cancels the forwarding that may have been previously specified for the BootP/DHCP requests that match the criteria.

Usage Notes

- Forwarding applies only to requests that are not excluded by the EXCLUDE subcommand.
- Forwarding specified for a gateway takes precedence over forwarding specified for an adapter.
- A hop count is maintained in the BOOT request. This hop count is incremented each time a DHCP daemon forwards the request. The DHCP server will not forward a request whose hop count is three or more.
- The BOOT request contains a server name field. This field allows the client to specify the host name of a DHCP daemon which should process the request. If the target DHCP daemon is not the receiving server, and the address of the server is not on the same cable as the adapter that received the request, then the request will be forwarded.

  Normally, a request is not forwarded to a target DHCP daemon when the target daemon is on the same cable as the adapter of the DHCP daemon that receives the original request. Such forwarding is not done because it is assumed that the target daemon would have heard this same request.

  However, you can override this and force a request to be sent to such a target daemon. If a FORWARD to a specific IP address is defined for a receiving adapter, requests will always be forwarded to the DHCP daemon at that IP address.
- Requests that are forwarded by a gateway contain a gateway IP address. The DHCP server specifies this gateway IP address as the address of the adapter which receives the original request. This allows the DHCP daemon, which ultimately builds the response packet for the requesting client, to send that packet to the correct gateway; that gateway then sends the response packet to the client.

  Only the DHCP daemon that initially hears the client request acts as the gateway; subsequent forwarding of the request by other DHCP daemons does not change the gateway IP address.
- Forwarding is useful if you wish to centralize your machine files on a specific host and have other DHCP daemons forward their received requests to the central site for processing. When you set up forwarding in this manner, you must take into account the increased load on the central server and the time required to forward requests. If the interval to respond to a client request is too long, that client may then retransmit its requests, and increase the network load.

  The following is a simple example of forwarding to central sites. The configuration consists of three hosts that run VM DHCP daemons:

  VMSAT1, VMSAT2
  Satellite VM hosts running DHCP daemons connected to subnets with clients that submit BootP/DHCP requests.

  VMMAIN Main VM host running DHCP for responding to BootP/DHCP requests from VMSAT1 and VMSAT2.

  In this example, requests received by the VMSAT1 or VMSAT2 are automatically routed to VMMAIN. This allows VMSAT1 and VMSAT2 to run DHCP daemons which do not maintain a *functional* machine file. A Forward statement would appear in the configuration files for VMSAT1 and VMSAT2 as:

  ```
  FORWARD ALL TO xxx.xxx.xxx.xxx ALWAYS
  ```

  where *xxx.xxx.xxx.xxx* is the IP address of VMMAIN.

- The FORWARD subcommand causes the DHCP daemon to query the current TCP/IP configuration of the host where the DHCP daemon is running, to determine the IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand's operation, whether the subcommand completes successfully or not.

HELP Subcommand

Use the HELP subcommand to display a brief description of available subcommands.

```
►►──HELP──────────────────────────────────────────────────────────────►◄
```

Operands

The HELP subcommand has no operands.

INCLUDE Subcommand

Use the INCLUDE subcommand to specify the adapter address or address of DHCP forwarding daemons for which requests should be handled.

```
                                        ┌─ANY──────┐
►►──INCLUDE──┬─ADApter──ipaddr─┬────────┤          ├──────────────────►◄
             └─ALL─────────────┘        ├─CLIents──┤
                                        ├─GATeways─┤
                                        └─ANY──────┘
```

Operands

ADApter *ipaddr*
: indicates the IP address of the adapter on the host system for which requests should be handled.

ALL
: indicates that requests from all adapters should be handled.

ANY
: indicates that any request that is received on the specified adapters (or "ALL") should be handled. This is the default.

CLIents
: indicates that only client BootP/DHCP requests that are received on the specified adapters (or "ALL") should be handled.

GATeways
: indicates that gateway-forwarded requests that are received on the specified adapters (or "ALL") should be handled.

Usage Notes

- The opposite of the INCLUDE subcommand is the EXCLUDE subcommand. Any values set by the INCLUDE subcommand may be reset by the EXCLUDE subcommand.
- The INCLUDE subcommand causes the DHCP daemon to query the current TCP/IP configuration of the host where the DHCP daemon is running, to determine the IP addresses defined for that host. Any included adapter that is not in the defined IP address list will be automatically excluded as the result of this subcommand's operation, whether the subcommand completes successfully or not.

LURK Subcommand

Use the LURK subcommand to toggle the LURK mode in the DHCP daemon. If the daemon is already operating in LURK mode, it will resume answering requests from clients. If the daemon is not operating in LURK mode, it will begin to listen for, but not will not respond to, client requests.

```
►►──LURK──────────────────────────────────────────────────────────►◄
```

Examples

1. The following is displayed on completion of this subcommand:

   ```
   LURK is now l
   ```

   where *l* is 0 if LURK mode is off; 1 if LURK mode is on.

QUIT Subcommand

Use the QUIT subcommand to stop the DHCP daemon. This subcommand is equivalent to the EXIT and STOP subcommands.

```
►►──QUIT──────────────────────────────────────────────────────────►◄
```

Operands

The QUIT subcommand has no operands.

RELOAD Subcommand

Use the RELOAD subcommand to reload the machine or configuration table file.

```
                                        (1)
           ┌─MACHINE──ETC──DHCPTAB──*──────────┐
►►──RELOAD─┼───────────────────────────────────┴──────────────────►◄
           │                              (1)  │
           │           ┌─ETC──DHCPTAB──*──────┐│
           ├─MACHINE───┼──────────────────────┴┤
           │           │                  (1) ││
           │           │     ┌─DHCPTAB *────┐ ││
           │           └─fn──┼──────────────┴─┘│
           │                 │          (1)  │ │
           │                 │      ┌─*───┐  │ │
           │                 └─ft───┼─────┴──┘ │
           │                        └─fm──┘    │
           │                               (1) │
           │          ┌─CNF──DHCPTAB──*──────┐ │
           └─CONFIG───┼──────────────────────┴─┘
                      │                (1)   │
                      │     ┌─DHCPTAB *────┐ │
                      └─fn──┼──────────────┴─┘
                            │          (1) │
                            │      ┌─*───┐ │
                            └─ft───┼─────┴─┘
                                   └─fm──┘
```

Notes:

1 If '*' is specified (or defaulted) for the file mode then the first file matching the file name and file type is used.

Operands

MACHINE
indicates that the file to be loaded is a machine file that contains client information.

fn is the file name of the file to load.

ft is the file type of the file to load.

fm is the file mode of the file to load.

CONFIG
indicates that the file to be loaded is a configuration file.

Usage Notes

- MACHINE and CONFIG are reserved keywords. They may not be used as file names or file types.
- The configuration file is composed of blank lines, comment lines, and configuration statements. Configuration statements have the same function and syntax as the DHCPD EXCLUDE, FORWARD, and INCLUDE subcommands.

  The format of the configuration file is:
  - One line per statement.
  - Blank lines are ignored.
  - Comment lines are ignored. Comment lines are lines where the first non-blank character is an "*", "#", or ";".

- The machine file is composed of blank lines, comment lines, entry lines for the clients, and tag control lines used to decipher the entry lines. The format and content of this file is described in the section “DHCPD Machine Statements” on page 286.
- When a configuration file is processed, all adapters are assumed to be included and no forwarding exists until specified otherwise by configuration file statements.

Examples

If the machine file is successfully loaded, the following response is displayed:

```
fn ft fm loaded in secs seconds.
Table reloaded from fn ft fm
DHCPD Ready;
```

where *fn*, *ft*, and *fm* are the respective file name, file type and file mode of the loaded file, and *secs* is the time required to load the file.

SHOW Subcommand

Use the SHOW subcommand to display information which would be returned to a client that specified parameters similar to those specified with the SHOW subcommand.

```
►►──SHOW──┬─ALL──────────────────────────────────────────────────┬────────►◄
          │           ┌───────────────────────────────────────┐  │
          │           ▼    (1)                                 │  │
          └─REQuest──────────────┬─ADDress──ipaddr──────────┬─┴──┘
                                 ├─CLIent──hwtype──clientid─┤
                                 ├─CLAss──class_name────────┤
                                 └─VENdor──vendor_name──────┘
```

Notes:

1 Each keyword string may be specified only once.

Operands

ALL
: indicates that the current client and configuration settings in the machine file should be displayed.

REQuest
: indicates that the operands which follow identify the client BootP/DHCP request. The output of this command should show the options and addresses that would be used if a client provided a request specifying specific client related options.

ADDress
: indicates that the operand which follows is an IP address from which the request would have been received. This can be the subnet address or a valid address on the subnet.

ipaddr
: is an IP address specified in dotted-decimal notation.

CLIent
: indicates that the two operands which follow specify a client ID.

hwtype
is the hardware type of the client computer, or 0. The valid client types are defined in STD 2, RFC 1700.

hwtype	Client hardware
1	ethernet ether
2	ethernet3 ether3
3	ax.25
4	pronet
5	chaos
6	token-ring tr
7	arcnet
8	hyperchannel
9	lanstar
10	autonet
11	localtalk
12	localnet
13	ultra_link
14	smds
15	frame_relay
16	atm
17	ieee802
18	fddi

clientid
is the hexadecimal MAC address or a name which identifies the client. If a name is specified then:
- *hwtype* must be 0
- If the name contains blanks, then it must be enclosed in single or double quotes.

CLAss
indicates that the operand that follows specifies a class name.

class_name
is the user-defined label that identifies the class. The client would specify the class name using option 77. 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 a pair of single quotes (') or a pair of double quotes (").

VENdor
indicates that the operand which follows specifies a vendor name.

vendor_name
is the user-defined label that identifies the vendor. The client would transmit this label using option 60. 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 a pair of single quotes (') or a pair of double quotes (").

Examples

1. If the "ALL" operand is specified, then output similar to the following is displayed:

```
SHOW ALL
GLOBAL DATA
  Boot Strap Server: 9.100.48.75
  Support BootP: Yes
  Support Unlisted Clients: No
  Lease Expire Interval: 1 MINUTES
  Lease Time Default: -1
  Ping Time: 1 SECONDS
  Reserved Time: 5 MINUTES
  Used IP Expire Interval: 30 SECONDS
  OPTION 4:  9.100.48.50
  VENDOR: IBM Network Station
    OPTION 43: /
  CLIENT: 6 0000e580fca8
    IP ADDRESS: ANY
    OPTION 4:  9.100.48.75
  CLASS: IBMNSM 1.0.0
    OPTION 67: /QIBM/ProdData/NetworkStation/kernel
SUBNET 9.100.57.0
  SUBNET MASK:  255.255.255.0
  RANGE: 9.100.57.1 - 9.100.57.99
  OPTION 3:  9.100.57.253
  OPTION 5:  9.100.25.252
  OPTION 6:  9.100.25.252
  OPTION 15: ibm.com
  CLASS: IBMNSM 1.0.1
    OPTION 67: /QIBM/ProdData/NetworkStation/me
SUBNET 9.100.57.0
  SUBNET MASK:  255.255.255.0
  RANGE: 9.100.57.100 - 9.100.57.140
  OPTION 3:  9.100.57.253
  OPTION 5:  9.100.25.252
  OPTION 6:  9.100.25.252
  OPTION 15: ibm.com
  CLASS: IBMNSM 1.0.1
    OPTION 67: /QIBM/ProdData/NetworkStation/altme
```

The Option lines in the output indicate the configuration options that are defined in the machine file. These options are sent to the client in the DHCP and BootP replies.

2. If the "REQUEST" operand is specified, then the response to a similar request (information that is the same as the specified operands) is shown. If there is more than one possible response then all of them are shown. This example uses the same machine file as the previous SHOW ALL example. In this example, the request would generate two possible responses; one from subnet 9.100.57.0 with address pool 9.100.57.1 to 9.100.57.99, and another from the same subnet with address pool 9.100.57.100 to 9.100.57.140.

```
SHOW REQUEST CLIENT 6 0000e580fca8 CLASS "IBMNSM 1.0.1"
SUBNET 9.100.57.0
  IP ADDRESS: 9.100.57.1 - 9.100.57.99
  Boot Strap Server: 9.100.48.75
  OPTION 3: 9.100.57.253
  OPTION 4: 9.100.48.75
  OPTION 5: 9.100.25.252
  OPTION 6: 9.100.25.252
  OPTION 15: ibm.com
  OPTION 67: /QIBM/ProdData/NetworkStation/me
SUBNET 9.100.57.0
  IP ADDRESS: 9.100.57.100 - 9.100.57.140
  Boot Strap Server: 9.100.48.75
  OPTION 3: 9.100.57.253
  OPTION 4: 9.100.48.75
  OPTION 5: 9.100.25.252
  OPTION 6: 9.100.25.252
  OPTION 15: ibm.com
  OPTION 67: /QIBM/ProdData/NetworkStation/altme
```

Note: The IP addresses shown do not denote currently available addresses but indicate the addresses which belong to the pool.

3. If the specified request will not be processed because an appropriate definition does not exist for the client in the machine file, the response will indicate this. This example uses the same machine file as the previous SHOW ALL example. In this example, address 9.200.55.55 does not correspond to any defined address pool.

```
SHOW REQUEST ADDRESS 9.200.55.55
  NO RESPONSE POSSIBLE
```

Usage Notes

- Addresses shown in the output of this command are possible IP addresses that might be used. They do not indicate whether the address is available. It is probable that many addresses may be currently leased out. To find the status of an address or a set of addresses, use the STATUS subcommand. See “STATUS Subcommand” for more information on the STATUS subcommand.

STATUS Subcommand

Use the STATUS subcommand to display the status of an IP address, Client, or Subnet’s addresses.

```
►►──STATUS──┬─CLIent──hwtype──clientid─┬──────────────────────────────────►◄
            ├─ADDress──ipaddr──────────┤
            └─SUBnet──ipaddr───────────┘
```

Operands

CLIent
: indicates that the two operands which follow specify a client ID. The status of the IP address currently leased to the client should be shown.

hwtype
: is the hardware type of the client computer, or 0. The valid client types are defined in STD 2, RFC 1700.

hwtype	**Client hardware**
1	ethernet ether
2	ethernet3 ether3
3	ax.25
4	pronet
5	chaos
6	token-ring tr
7	arcnet
8	hyperchannel
9	lanstar
10	autonet
11	localtalk
12	localnet
13	ultra_link
14	smds
15	frame_relay
16	atm
17	ieee802
18	fddi

clientid
is the hexadecimal MAC address or a name which identifies the client. If a name is specified then:
- *hwtype* must be 0
- If the name contains blanks, then it must be enclosed in single or double quotes.

ADDress
indicates that the operand which follows is an IP address whose status should be shown.

ipaddr
is an IP address specified in dotted-decimal notation.

SUBnet
indicates that the operand which follows is an IP address of a subnet. The status of the IP addresses associated with the subnet should be shown.

Examples

1. If a specific address is being queried by either the CLIENT or ADDRESS operand, then the following would be shown:
   - If the address is available for reassignment:
     ```
     STATUS ADDRESS 9.100.57.110
     9.100.57.110    AVAILABLE
     ```
   - If the lease for this address has expired, but the address is waiting to reenter the available pool:
     ```
     STATUS ADDRESS 9.100.57.111
     9.100.57.111    EXPIRED. HELD UNTIL 23:59:00 ON 24 DECEMBER 1997
                     TO: 0 BEACH
     ```
   - If an address is currently leased to a client:
     ```
     STATUS ADDRESS 9.100.57.112
     9.100.57.112    IN USE UNTIL 23:59:00 ON 24 DECEMBER 1997
                     TO: 0 STEVEGESSNER
     ```
   - If an address is permanently leased to a client:
     ```
     STATUS ADDRESS 9.100.57.113
     9.100.57.113    IN USE PERMANENTLY
                     TO: 0 ADAMGESSNER
     ```
   - If an address is being pinged before a DHCP response is sent:
     ```
     STATUS ADDRESS 9.100.57.114
     9.100.57.114    WAITING FOR ICMP REPLY BEFORE BEING OFFERED
                     TO: 0 PAULAGESSNER
     ```
   - If an address is being offered to a client:
     ```
     9.100.57.115    BEING OFFERED
                     TO: 0 MARK
     ```
   - If an address was excluded or not specified in the machine file:
     ```
     STATUS ADDRESS 9.100.57.100
     9.100.57.110    EXCLUDED
     ```
2. If a pool of addresses is being queried, then the output will provide information for each IP address that belongs to the pool, and which is not excluded. In the following example, subnet 9.100.57.0 has 6 addresses defined from 110 to 115.
   ```
   STATUS SUBNET 9.100.57.0
   9.100.57.110    AVAILABLE
   9.100.57.111    EXPIRED. HELD UNTIL 23:59:00 ON 24 DECEMBER 1997
                   TO: 0 BEACH
   9.100.57.112    IN USE UNTIL 23:59:00 ON 24 DECEMBER 1997
   ```

```
                      TO: 0 STEVEGESSNER
    9.100.57.113      IN USE PERMANENTLY
                      TO: 0 ADAMGESSNER
    9.100.57.114      WAITING FOR ICMP REPLY BEFORE BEING OFFERED
                      TO: 0 PAULAGESSNER
    9.100.57.115      BEING OFFERED
                      TO: 0 MARK
```

STAYUP Subcommand

Use the STAYUP subcommand to toggle the STAYUP mode in the DHCP daemon. If the daemon is already operating in STAYUP mode, it will cease operating in this mode and will end processing if a subsequent TCP/IP failure occurs. If the daemon is not operating in STAYUP mode, it will begin to ensure processing will not end if a subsequent TCP/IP failure occurs.

```
►►──STAYUP──────────────────────────────────────────────────────────►◄
```

Examples

1. The following is displayed on completion of the subcommand:

   ```
   STAYUP is now s
   ```

   where *s* is 0 if STAYUP mode is off; 1 if STAYUP mode is on.

Usage Notes

- The STAYUP subcommand is needed only when the TCP/IP machine does not contain an entry for the virtual machine running DHCPD.

STOP Subcommand

Use the STOP subcommand to stop the DHCP daemon. This subcommand is equivalent to the EXIT and QUIT subcommands.

```
►►──STOP────────────────────────────────────────────────────────────►◄
```

Operands

The STOP subcommand has no operands.

TRACE Subcommand

Use the TRACE subcommand to toggle the TRACE mode in the DHCP daemon. If the daemon is already operating in TRACE mode, it will cease displaying debug information as it processes requests. If the daemon is not operating in TRACE mode, it will display debug information as it processes requests.

```
►►──TRACE───────────────────────────────────────────────────────────►◄
```

Examples

1. The following is displayed on completion of this subcommand:

   ```
   TRACE is now t
   ```

   where *t* is 0 if TRACE mode is off; 1 if TRACE mode is on.
2. See *TCP/IP Programmer's Reference* for a description of DHCP server trace output.

Chapter 12. Configuring the FTP Server

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

FTP Server Configuration Steps

1. Update the TCP/IP server configuration file.
2. Update the DTCPARMS file for the FTP server.
3. Establish FTP server machine authorizations.
4. Customize the SRVRFTP CONFIG file.
5. Configure Automatic File Translation. (Optional)
6. Customize FTP server exits. (Optional)

Dynamic Server Operation

The FTP server provides a VM Special Message (SMSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information, see "SMSG Interface to the FTP Server" on page 349.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  FTPSERVE 0
```

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

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

Step 2: Update the DTCPARMS File

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

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

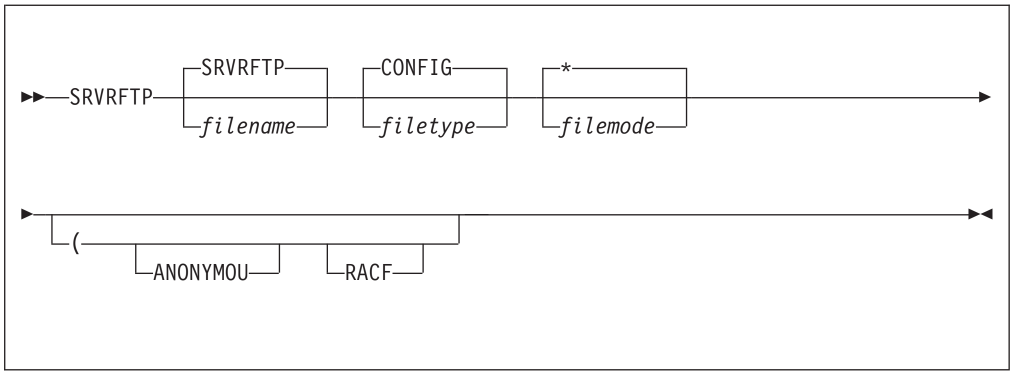
If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the FTP server if you:
- use a configuration file other than SRVRFTP CONFIG
- provide anonymous FTP support
- use an ESM for client authorization and access control.

SRVRFTP Command

FTP services are initiated using the SRVRFTP command:



Specify SRVRFTP command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

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

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

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

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

RACF
: Directs the FTP server to rely upon an External Security Manager (ESM) to validate passwords and to control access to minidisks. This operand is automatically supplied when an `ESM_Enable.YES` entry is specified in the DTCPARMS file.

Step 3: Establish FTP Server Machine Authorizations

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

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

```
POSIXINFO UID 0 GID 0
```

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

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

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

The FTP server can use an external security manager (ESM) to authenticate FTP clients and to control access to z/VM resources. To use an ESM, specify `:ESM_Enable.Yes` in the DTCPARMS file. For more information, see "Appendix A. Using TCP/IP with an External Security Manager" on page 617.

| Step 4: Customize the SRVRFTP CONFIG File

| The FTP configuration file, SRVRFTP CONFIG, defines how the FTP server is to
| operate and what services and types of access it provides. See "FTP Server
| Configuration File Statements" for detailed information about how to specify
| entries within this file. A sample FTP configuration file is shipped with TCP/IP
| Function Level 330 as SRVRFTP SCONFIG on the TCPMAINT 591 disk. Your
| customized SRVRFTP CONFIG file should be stored on the TCPMAINT 198
| minidisk.

| FTP Server Configuration File Statements

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

| ANONYMOU Statement

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

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

```
►►──ANONYMOU──────────────────────────────────────────────────────────►◄
```

Operands

The ANONYMOU statement has no operands.

Usage Notes

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

AUTOTRANS Statement

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

```
     ┌─AUTOTRANS OFF──────┐
►►───┴─AUTOTRANS──┬─ON──┬─┴───────────────────────────────────────────►◄
                  └─OFF─┘
```

Operands

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

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

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

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

DONTREDIRECT Statement

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

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

```
►►──DONTREDIRECT──────────────────────────────────────────────────────────►◄
```

Operands

The DONTREDIRECT statement has no operands.

FTAUDIT Statement

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

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

```
►►──FTAUDIT───────────────────────────────────────────────────────────────►◄
```

Operands

The FTAUDIT statement has no operands.

FTCHKCMD Statement

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

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

```
►►──FTCHKCMD──────────────────────────────────────────────────────────────►◄
```

Operands

The FTCHKCMD statement has no operands.

FTCHKDIR Statement

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

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

```
►►──FTCHKDIR─────────────────────────────────────────────────────────►◄
```

Operands

The FTCHKDIR statement has no operands.

INACTIVE Statement

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

```
    ┌─INACTIVE 300──────┐
►►──┼───────────────────┼────────────────────────────────────────────►◄
    └─INACTIVE──seconds─┘
```

Operands

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

LISTFORMAT Statement

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

```
    ┌─LISTFORMAT VM────────┐
►►──┼──────────────────────┼─────────────────────────────────────────►◄
    └─LISTFORMAT──┬─VM───┬─┘
                  └─UNIX─┘
```

Operands

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

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

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

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

For detailed information about VM-format and Unix-format responses, see the *TCP/IP User's Guide*.

LOADDBCSTABLE Statement

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

```
►►──LOADDBCSTABLE──┬─EUCKANJI──┬──────────────────────────────────────►◄
                   ├─HANGEUL───┤
                   ├─JIS78KJ───┤
                   ├─JIS83KJ───┤
                   ├─KSC5601───┤
                   ├─SJISKANJI─┤
                   └─TCHINESE──┘
```

Operands

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

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

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

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

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

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

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

Usage Notes

- Additional virtual storage may need to be defined for the FTP server if a large number of translate tables are loaded concurrently.
- The IBMKANJI transfer type does not require any translate table to be loaded. For more information on loading and customizing DBCS translate tables, see "Customizing DBCS Translation Tables" on page 601.

PORT Statement

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

```
   ┌─PORT 21────────────┐
►►─┼────────────────────┼──────────────────────────────────────►◄
   └─PORT──port_number──┘
```

Operands

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

RACF Statement

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

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

For more information about using an external security manager, see "Appendix A. Using TCP/IP with an External Security Manager" on page 617.

```
►►──RACF───────────────────────────────────────────────────────►◄
```

The RACF statement has no operands.

Usage Notes

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

RDR Statement

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

```
►►──RDR──filemode──────────────────────────────────────────────────────►◄
```

Operands

filemode
: The file mode of the resource to be used to temporarily store file before they are sent to a virtual reader. Any of the following resources may be used for this purpose:
  - a minidisk
  - a temporary minidisk
  - a virtual disk
  - an SFS directory.

Usage Notes

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

  For example, assume you choose to use the FTP server "root" SFS directory in file pool MYFPOOL1 as the storage area for files directed to a user's virtual reader. The following DTCPARMS entry (added to the appropriate FTP *server* or *class* definition) will configure the FTP server to acquire this resource with file mode F:

  ```
  :nick.FTPSERVE
    :vmlink. .DIR MYFPOOL1:FTPSERVE. <* F>
  ```

  Additionally, a corresponding **RDR F** statement must be included in the FTP server configuration file. File mode F is used here to ensure the MYFPOOL1:FTPSERVE. directory is accessed in the CMS search order after the TCP/IP configuration, client-code, and server-code minidisks.

TRACE Statement

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

```
          ┌─CONSOLE──┐
►►──TRACE─┼──────────┼────────────────────────────────────────────────►◄
          ├─CONSOLE──┤
          └─FILE─────┘
```

Operands

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

FILE
Specifies that trace information should be directed to the FILE DEBUGTRA file on the FTP server 191 minidisk.

Step 5: Configure Automatic File Translation (Optional)

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

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

To enable default automatic file translation:
1. Specify the AUTOTRANS ON statement in the FTP server configuration file.
2. Customize the TCPIP DATA file to include the appropriate VMFILETYPE and VMFILETYPEDEFAULT statements. The FTP server relies upon these statements to control the manner in which file translation is performed for specific file extensions, as well as those that are "unknown" or not recognized. For detailed information about how to specify these statements, see "Chapter 5. Defining the TCP/IP System Parameters" on page 139.

Notes:
1. The VMFILETYPE statement determines whether EBCDIC-ASCII translation occurs for a specific file, based on the extension of that file.
2. File extensions that are not dealt with through a specific VMFILETYPE statement are considered as "unknown" (that is, are not recognized). The translation performed for such files (if any) is controlled by the VMFILETYPEDEFAULT statement. If the VMFILETYPEDEFAULT statement is not used, unknown file extensions default to a transfer type of **Image** and no translation is performed.
3. The FTP server ignores the LINES operand of the VMFILETYPE and VMFILETYPEDEFAULT statements. For additional information about automatic file translation, see the *TCP/IP User's Guide*.

Dynamic Server Operation

The VM Special Message Facility (SMSG) command provides an interactive interface to the FTP server to perform privileged system administration tasks, such as:
- enabling and disabling the Trace function
- querying user data
- dropping connections
- querying minidisks and directories held by the FTP server

- enabling, disabling, and querying the FTP User Exits
- setting the default list format supplied by the server
- enabling and disabling default automatic file translation.

Notes:

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

SMSG Interface to the FTP Server

```
>>──SMSG──server_id──┬─process_name──┬─────┬──────┬──────────────────><
                     │               ├─ON──┤      │
                     │               └─OFF─┘      │
                     │             ┌─ON──┐        │
                     ├─AUTOTrans───┼─────┼────────┤
                     │             └─OFF─┘        │
                     ├─CLosecon───────────────────┤
                     ├─DRop──conn_num─────────────┤
                     ├─FTPEXIT──┬────────┬────────┤
                     │          └─RELOAD─┘        │
                     ├─Help───────────────────────┤
                     ├─LISTFormat──┬─VM───┬───────┤
                     │             └─UNIX─┘       │
                     │       ┌─*────────────┐     │
                     ├─Query─┼──────────────┼─────┤
                     │       ├─user_id──────┤     │
                     │       ├─process_name─┤     │
                     │       ├─ACCESSED─────┤     │
                     │       ├─AUTOTrans────┤     │
                     │       └─LISTFormat───┘     │
                     ├─REBoot─────────────────────┤
                     ├─REFresh VMFiletype─────────┤
                     ├─RELease──fm────────────────┤
                     ├─SHutdown───────────────────┤
                     │       ┌─ON CONsole───────┐ │
                     └─TRace─┼──────────────────┼─┘
                             │    ┌─CONsole─┐   │
                             ├─ON─┼─────────┼───┤
                             │    └─FIle────┘   │
                             └─OFF──────────────┘
```

Operands

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

AUTOTrans ON | **OFF**
: Determines whether automatic file translation is performed by default when files are transferred using the **Image** transfer type. Specify **ON** if automatic file translation should be enabled as the default when an FTP session is

established, or **OFF** if the default should be to not attempt such translation. If neither **ON** or **OFF** is specified, **ON** is the default.

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

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

For more information about automatic file translation, see "Step 5: Configure Automatic File Translation (Optional)" on page 348.

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

server_id
Specifies the user ID of the FTP server virtual machine.

DRop *conn_num*
Indicates the FTP server is to drop the specified connection.

FTPEXIT RELOAD
Reloads the FTP exit routine.

Help
Provides brief help about SMSG commands supported by the FTP server.

LISTFormat VM | UNIX
Sets the format default for list information supplied by the server when it responds to client DIR (or, LIST) requests. Specify **VM** for VM-format lists to be supplied by default, or **UNIX** if the default should be Unix-format lists. The specified format default is applied to all LIST responses, unless:

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

For detailed information about VM-format and Unix-format responses, see the *TCP/IP User's Guide*.

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

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

Query ACCESSED
Returns CMS QUERY ACCESSED command output.

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

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

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

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

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

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

TRace OFF
Disables server tracing.

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

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

Providing Web Browser FTP Support

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

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

In addition, automatic file translation may need to be enabled on a default basis, to allow for correct file translations when web browser and graphical FTP clients are in use. This is because many such browsers often default to using a transfer type of **Image** (or, **binary**) and do not offer a way for users to specify a different file transfer type. See “Step 5: Configure Automatic File Translation (Optional)” on page 348 for more information about this topic.

Step 6: Customize FTP Server Exits (Optional)

The FTP server exits are described in the following sections.

Using the FTP Welcome Banner

The FTP server can display a site specific message when users establish a connection to the FTP server. The contents of file "FTP BANNER" will be displayed, if the file exists. When this file exists, the first such file found in the

CMS search order is used. You may also wish to have one of several different banners displayed after the user name and login password (for other than anonymous user) validation. This is handled in the CHKIPADR Exit.

Using the FTP Server Exit

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

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

When the FTP server exit is enabled for general command exit processing (FTCHKCMD enabled), the FTP exit will be called to perform command validation for every received FTP command. The general command exit can be used to perform additional security checking and then take an appropriate action, such as:
- Reject commands from a particular IP address or user ID
- Reject a subset of commands for anonymous users
- Reject transfer requests for specific files
- Reject all store (APPE, STOR, STOU) commands supplied by users.

With the FTP exit enabled for CD command exit processing (FTCHKDIR enabled), the exit can validate FTP directory changes and provide greater control over access to system resources by selectively honoring or refusing a client change directory request. The exit is called when an FTP client provides one of the following commands:
- CWD or CD, to change the working directory
- CDUP, to change the working directory to the parent directory
- PASS, provided a default directory is defined in CHKIPADR EXEC for the user that supplies this command
- USER, for an anonymous login for which a default directory is defined in CHKIPADR EXEC
- APPE, DELE, LIST, NLST, RETR, SIZE, STOR, or STOU commands that involve an explicit change in directory.

Sample FTPEXIT exec and assemble routines are supplied as softcopy files (FTPEXIT SEXEC and FTPEXIT SAMPASM, respectively) on the TCPMAINT 591 minidisk. Refer to the *TCP/IP Programmer's Reference* for details about FTPEXIT parameter list and parameter descriptions.

Using the CHKIPADR Exit

The CHKIPADR exit provides a means for controlling several aspects of FTP server processing at the time an FTP connection is initiated by the user. This capability is provided through the CHKIPADR EXEC, which is invoked by the server each time a user logs in. This exec may be used to:
- Permit or deny client access to FTP services
- Permit anonymous user login for users other than ANONYMOU
- Select a default working directory

- Select a “welcome” message, or banner
- Select a default file list format
- Select default automatic file translation

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

Providing Anonymous Login Support

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

Establishing a Default Working Directory

When a user logs in using FTP, the 191 minidisk associated with that user ID is established as a working directory, by default. However, an alternate working directory may be selected for a user when a connection is established. The alternate working directory specified may be a:
- minidisk
- Shared File System (SFS) directory
- Byte File System (BFS) directory
- virtual reader (RDR).

Providing a User-Specific Banner

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

Establishing a Default File List Format

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

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

Establishing Automatic File Translation Defaults

The default setting for automatic file translation, based on file extension, can be turned on or off for specific users. For detailed information, see “Step 5: Configure Automatic File Translation (Optional)” on page 348. The desired default setting must be indicated when control is returned to the FTP server.

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

CHKIPADR Input

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

```
►►──CHKIPADR──userid──ipaddress──byuserid──────────────────────────────────►◄
```

Operands

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

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

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

CHKIPADR Output

These are the return codes for the CHKIPADR EXEC:

0	User ID/IP Address is authorized
4	User ID is not authorized
8	IP Address is not authorized
12	User ID is not authorized and no error message will be sent user.
20	Anonymous user, no password required

Program stack contents upon exit may contain in any order:

- Default working directory
- Banner file name, prefixed by the keyword **BANNER**
- Translation default (**ON** or **OFF**), prefixed by the keyword **AUTOTRANS**
- List format (**VM** or **UNIX**), prefixed by the keyword **LISTFORMAT**

Example

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

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

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

```
      Queue 'autotrans on'
      status = 20
   End
```

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

```
ftp vmsys1
VM TCP/IP FTP Level 330
Connecting to VMSYS1 9.130.48.64
220-........................................
220-.                                      .
220-.  This is the contents of FTP BANNER      .
220-.                                      .
220-........................................
220-FTPSERV2 IBM VM Level 330 at VMSYS1.ENDICOTT.IBM.COM, 13:56:24 EST
2000-03-26
220 Connection will close if idle for more than 5 minutes.
USER (identify yourself to the host):
teri
>>>USER teri
230-........................................
230-.                                      .
230-.  This is the contents of WELCOME BANNER  .
230-.                                      .
230-........................................
230 TERI logged in; working directory = SERVER1:TERI.FTP
Command:
```

Chapter 13. Configuring the Kerberos Server

This chapter describes how to configure and customize the Kerberos Authentication System for TCP/IP.

The Kerberos system in TCP/IP consists of the following:
- Kerberos database
- Kerberos authentication server
- Remote database administrator server
- Local database administrator utilities
- Remote database administrator utility
- Client utilities.

The Kerberos authentication server provides a way for authenticated users to prove their identity to other servers in a network. The authentication server uses the Kerberos database to verify that the client making the request is, in fact, the client named in the request. The authentication server runs in the VMKERB virtual machine.

The Kerberos remote database administration server allows you to modify the Kerberos database remotely using the KADMIN command. The Kerberos administration server runs in the ADMSERV virtual machine.

Local Kerberos database functions, such as creating or deleting the database, are performed by stopping the administration server and issuing the necessary commands at the CMS command line.

All changes made to the Kerberos database are made by the ADMSERV virtual machine because the database resides on a CMS minidisk and only one virtual machine can have write access to the minidisk at any one time. The authentication server accesses the Kerberos database in read mode.

Utilities available to Kerberos clients are described in *TCP/IP User's Guide*.

For more information, about the application programming interface, see *TCP/IP Programmer's Reference*.

Kerberos Name Structures

Before you customize your Kerberos Authentication System, you should be familiar with the structure of a Kerberos name. A Kerberos name consists of the following three parts.

Name	Description
principal name	Specifies the unique name of a user (client) or service.
instance	Indicates a label that is used to distinguish among the variations of the *principal name*. An *instance* allows for the possibility that the same client or service can exist in several forms that require distinct authentication.

	For users, an *instance* can provide different identifiers for different privileges. For example, the `admin` *instance* provides special privileges to the users assigned to it. For services, an *instance* usually specifies the host name of the machine that provides the service.
realm	Specifies the name of an administrative entity. The *realm* identifies each independent Kerberos site. The *principal name* and *instance* are qualified by the realm to which they belong, and are unique only within that realm. The *realm* is commonly the domain name. **Note:** You must express the *realm* as a name rather than an address number. For example, use endicott.ibm.com rather than 9.130.0.0.

Configuring the Authentication and Remote Administration Servers

Kerberos Servers Configuration Steps:

1. Update PROFILE TCPIP
2. Define the Kerberos services in ETC SERVICES
3. Update DTCPARMS for the authentication server
4. Update DTCPARMS for the remote database administration
5. Create and update the Kerberos system files
6. Build the Kerberos database
7. Store the master key
8. Start the Kerberos servers

Step 1: Update PROFILE TCPIP

Include VMKERB and ADMSERV in the AUTOLOG statement to automatically start the VMKERB and ADMSERV virtual machines when TCPIP is invoked. Verify that the following statements have been added to PROFILE TCPIP.

```
AUTOLOG
  VMKERB   0
  ADMSERV  0
```

Kerberos requires that ports 750 and 751 be reserved for it. Port 750 is used by the authentication server. Port 751 is used by the administration server. The following port entries must be in PROFILE TCPIP:

```
PORT
  750 TCP VMKERB
  750 UDP VMKERB
  751 TCP ADMSERV
  751 UDP ADMSERV
```

Step 2: Update ETC SERVICES

Before you set up the Kerberos system, you must define the Kerberos services. Verify that the following lines have been added to the ETC SERVICES file on the client code disk, TCPMAINT 592:

```
kerberos            750/tcp
kerberos            750/udp
kerberos_master     751/tcp
kerberos_master     751/udp
sample              906/udp
sample              906/tcp
```

You should compare this list to the port assignments in the PROFILE TCPIP file to ensure that there are no conflicts with any other service.

The entries for service `sample` enable you to run the Kerberos verification programs.

Step 3: Update DTCPARMS for the Authentication Server

The default TCP/IP configuration runs the Kerberos authentication server in the VMKERB virtual machine. The TCP/IP server initialization program searches for the configuration definitions for each server.

Note: You should modify the DTCPARMS file for the authentication server if you:
- Change the parameters passed to the VMKERB command.
- Change the user ID of the virtual machine that receives the VMKERB console log.

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Step 4: Update DTCPARMS for the Administration Server

The default TCP/IP configuration runs the Kerberos administration server in the ADMSERV virtual machine. The DTCPARMS file searches for the server configuration definitions for each server.

Note: You should modify the DTCPARMS file for the administration server if you:
- Change the parameters passed to the ADM_SERV command.
- Change the user ID of the virtual machine that receives the ADM_SERV console log.

If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used. For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Step 5: Create and Update the Kerberos System Files

This step describes the files that you must create and update for the Kerberos system.

Kerberos Configuration File

The Kerberos configuration file KRB CONF identifies hosts that are running the Kerberos authentication server.

The format of the file is:

```
realm
realm host_name
realm host_name admin server
```

The contents of the KRB CONF file are case-sensitive.

Some examples of the file format are:

```
univ.educ.chem
univ.other.dept joanpc
univ.educ.math chrispc admin server
```

The first line defines the local *realm* to which this VM host belongs. Each subsequent line specifies a remote realm and the host where the Kerberos server is running in that realm.

In our example, the third line lists `admin server` to indicate that the host provides a remote administration database server. The *host_name* that you specify must also be defined in your Domain Name Server or in the HOSTS LOCAL file.

The KRB CONF file must be made available to client applications. A sample of this configuration file is provided as KRB SCONFIG on the TCPMAINT 592 minidisk. Your customized configuration file should be copied and maintained on this same minidisk with a file identifier of KRB CONF.

Note: You cannot use dotted decimal IP addresses, such as 9.130.57.21, in the KRB CONF file.

Kerberos Remote Administrator Authorization Files

To authorize the remote database administrator to add, view, or modify database entries, you must create several remote administrator authorization files. These files, and their corresponding sample equivalents, are:

ADM@ACL ADD	Lists remote administrators that may add new principals. This file is provided in sample form as ADM@AADD SAMPAUTH.
ADM@ACL GET	Lists remote administrators that may view principal entries. This file is provided in sample form as ADM@AGET SAMPAUTH.
ADM@ACL MOD	Lists remote administrators that may change a principal's password. This file is provided in sample form as ADM@AMOD SAMPAUTH.

The sample authorization files are provided on the TCPMAINT 591 minidisk. However, your customized authorization files must be accessed by the remote database administration server (ADMSERV); copy and maintain your customized files on the ADMSERV 191 disk.

Your Kerberos principal name must be in these files if you want to use the KADMIN command to remotely administer the Kerberos databases. Without an entry in one of these files, you will only be able to change your `admin` instance password.

The format of these files is:

```
administrator's_principal_name.admin@realm
```

The instance must be **admin** and *realm* is usually the domain name. The contents of these files are case-sensitive.

These files can contain multiple entries in the same format.

Example: `tcpmaint.admin@univ.educ.chem`

Step 6: Build the Kerberos Database

You must enter Kerberos commands from the ADMSERV user ID to create and use the Kerberos database before you can start the Kerberos remote 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 files.
2. The system prompts 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 KRB CONF file. The default is `YOUR_KRB.RE.ALM`.
3. The system prompts you for the master key.
4. At the prompt, enter the master key.

   You need the master key to manage the Kerberos database. If the Kerberos database file already exists, the system indicates that the file already exists. Use KDB_DEST to destroy the existing database file. See "KDB_DEST Command" on page 369 for more information.

KDB_INIT will create two database files:

- PRINCPL DAT
- PRINCPL IDX

See "KDB_INIT Command" on page 371 for more information.

Step 7: Store the Master Password

The following steps describe how to store the master key so that the Kerberos authentication server will not have to prompt you for the master password.

1. Issue the KSTASH command.

   The system prompts you for the master password.
2. Enter the master password at the password prompt.

See "KSTASH Command" on page 372 for more information.

Step 8: 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

Autolog or log on user ID VMKERB. The Kerberos server will start.

Maintaining the Log: When the authentication server starts, it creates a log file called KERBEROS LOG. All transactions to the Kerberos authentication server are recorded in this file. Because Kerberos appends to the file continuously, you should periodically monitor the size of the KERBEROS LOG file, erasing or editing the log if necessary.

Stopping the Authentication Server: The authentication server must be stopped manually using the HX command.

Start the Kerberos Remote Administration Server

Autolog or log on user ID ADMSERV. The remote administration server will start.

Once 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 367 for more information about the KADMIN command.

Maintaining the Log: When the remote database administration server starts, it creates a log file called ADM_SERV SYSLOG. All transactions to this server are recorded in this file. Because Kerberos appends to the file continuously, you should periodically monitor the size of the ADM_SERV SYSLOG file, erasing or editing the log if necessary.

Stopping the Remote Database Administration Server: The remote database authentication server must be stopped manually using the HX command.

Setting Up a Kerberos Service or Client Application

Kerberos services and client applications can be developed by referencing the | SAMPLE@S and SAMPLE@C programs, which are provided on the client code | disk, TCPMAINT 592.

Setting Up a Kerberos Service Application

Use the following steps to set up a service application that uses Kerberos functions:

1. Register the service with the Kerberos database locally using KDB_EDIT or remotely using the KADMIN function. The service name is used as the principal and the host name where the service is running is used as the instance. For example, a Kerberos-enabled 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.
2. After you register all the services provided by the same host, enter the command:

   ```
   EXT_SRVT host_name
   ```

   For example, `EXT_SRVT chrispc` generates the CHRISPC SRVTAB file. The server's keys will be stored in this file.
3. Transfer the key file to the host providing the services (chrispc in our example).
   - If the host providing the services is VM, copy the key file to the client code disk, TCPMAINT 592, as ETC SRVTAB.
   - If the host providing the services is MVS, copy the key file to *service_id*.ETC.SRVTAB,

- If the host providing the services is OS/2®, DOS, UNIX, or AIX, copy the key file to SRVTAB in the ETC directory.

4. Define the service name and assign port numbers in the ETC SERVICES file on the client code disk, TCPMAINT 592.
5. Start the service program on the host providing the service.

Setting Up a Client Application

The following steps describe how to set up a client application that uses Kerberos functions:

1. The database administrator should register the client with the Kerberos database locally using KDB_EDIT or remotely using the KADMIN function.
2. Verify that the KRB CONF file on the client code disk, TCPMAINT 592, contains a valid entry. For more information see "Step 5: Create and Update the Kerberos System Files" on page 359.
3. Issue the KINIT command to get an initial ticket. The ticket is saved in the TMP TKT0 file.
4. Start the Kerberos client application.

Use the KLIST command to see the ticket in the client's ticket file. The TMP TKT0 file contains tickets used by client applications for different servers.

You can use the KDESTROY command to delete the ticket file.

See *TCP/IP User's Guide* for the format and description of the parameters of the KINIT, KLIST, and KDESTROY commands.

Verifying the Kerberos Configuration

TCP/IP Level 3A0 provides a sample application client program (SAMPLE_C) and a sample application server program (SAMPLE_S) 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 file for the sample service
4. Transfer the service key file to the server
5. Start the sample server
6. Get the initial ticket
7. Run the sample client program

Two additional user IDs are required on your VM system for verification of Kerberos. One user ID will run the sample server, the other the sample client. Both must have access to the client code disk, TCPMAINT 592, and have the C language runtime library globaled.

In this example, the name *vm_host* is used to represent the name of your VM host.

Step 1: Set Up the Environment

Follow steps 1 through 8 in the configuration process.

Start the Kerberos authentication server as described in “Start the Kerberos Authentication Server” on page 361.

Note: If you fail to start the authentication server, error messages will be seen in the output to the console.

Step 2: Register the Sample Service and the User

The following steps describe how to register the sample service and user with Kerberos. The sample client program assumes that the service instance name is the name of the host that is running the sample service application.

Log on to ADMSERV, stop the remote administration server if it is running, and issue the KDB_EDIT command. The prompts and responses are in Table 22.

Table 22. Adding a Service and a Client to the Kerberos Database

Prompt	Response
Enter Kerberos master key:	krbpass
Principal name:	sample
Instance:	*vm_host*
<not found>, Create [y] ?	<enter>
New Password:	sam
Verifying, please re-enter New Password:	sam
Expiration date?	<enter>
Max ticket lifetime?	<enter>
Attributes?	<enter>
Edit O.K.	
Principal name:	user1
Instance:	<enter>
<not found>, Create [y] ?	<enter>
New Password:	use
Verifying, please re-enter New Password:	use
Expiration date?	<enter>
Max ticket lifetime?	<enter>
Attributes?	<enter>
Edit O.K.	
Principal name:	<enter>

See “KDB_EDIT Command” on page 369 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 File for the Sample Service

The following steps describe how to generate the key file for the sample service.

1. On the ADMSERV user ID, enter `EXT_SRVT` *vm_host*

   The system prompts you for a password.
2. Enter `krbpass` at the password prompt.
3. A key file *vm_host* SRVTAB is created that contains the keys for all services that use Kerberos on your VM system.

   See "EXT_SRVT Command" on page 366 for more information about the EXT_SRVT command.

Step 4: Transfer the Key File to the Server

Copy the *vm_host* SRVTAB to the client code disk, TCPMAINT 592, as ETC SRVTAB.

Step 5: Start the Sample Server

The following describes how to start the sample server, SAMPLE_S.

1. Log on to the sample server user ID.
2. Access the client code disk, TCPMAINT 592.
3. Global the C runtime library.
4. Run SAMPLE_S to start the sample Kerberos service application.

Note: SAMPLE_S will run until stopped with HX.

Step 6: Get the Initial Ticket

The following steps describe how you get the initial ticket from the Kerberos authentication server.

1. Log on to the sample client user ID.
2. Access the client code disk, TCPMAINT 592.
3. Global the C runtime library.
4. Issue the KINIT command to get the initial ticket.

   The system prompts you for your Kerberos name.
5. Enter `user1` at the Kerberos name prompt.

   The system prompts you for a password.
6. Enter `use` at the password prompt.

   If the KINIT command is successful, the TMP TKT0 ticket file is created. If the KINIT command is not successful an error message is issued.

You can use the KLIST command to view the initial ticket.

See the *TCP/IP User's Guide* for the format and description of the parameters of the KINIT and KLIST commands.

Step 7: Run the Sample Client Program

To start the sample Kerberos client program, enter

```
SAMPLE_C vm_host 100
```

on the sample client user ID.

Kerberos Server

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.
```

Administrative Commands for the Kerberos Database

Table 23 shows the Kerberos commands you can use to create and administer a Kerberos database.

Table 23. Summary of Kerberos Database Commands

Command	Description	Page
EXT_SRVT	Generates key files for specified instances from ADMSERV user ID	366
KADMIN	Adds, retrieves, or modifies the Kerberos database remotely from any ID	367
KDB_DEST	Erases Kerberos database files from ADMSERV user ID	369
KDB_EDIT	Registers users to the Kerberos database from ADMSERV user ID	369
KDB_INIT	Builds and formats the Kerberos database from ADMSERV user ID	371
KDB_UTIL	Loads or dumps the Kerberos database from ADMSERV user ID	371
KSTASH	Stores the master key	372

EXT_SRVT Command

Use the EXT_SRVT command to generate a key file for instances.

A key file 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 files are used by the servers to verify whether a ticket presented by a user is legal.

You should rename this file 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.

```
               ┌─────────────┐
               ▼             │
►►──EXT_SRVT────instance─────┴──────────────────────────────────────►◄
```

Operands

instance
The host name for which a key file 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 file INST1 SRVTAB is generated.

```
EXT_SRVTINST1 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 files.
- You can use the KLIST -srvtab command to see the contents of a key file. See *TCP/IP User’s Guide* for the usage of the KLIST command.
- Copy each *instance* SRVTAB key file to the corresponding instance’s host, and rename the key file as required by the host. See “Setting Up a Kerberos Service Application” on page 362 for more information on the required naming.
- If the *instance* name is longer than eight characters, the file name of the SRVTAB file generated is the first eight characters of the *instance* name. If the *instance* name results in a file name that is not valid for CMS, the file name TMP is used.

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.

```
►►──KADMIN──────────────────────────────────────────────────►◄
```

Operands

The KADMIN command has no operands.

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, ADMSERV, 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 ADMSERV user ID. Make the Kerberos instance be `admin`.
3. Have your Kerberos name in the remote administrator authorization files to perform the corresponding database operations.

- You can use the following subcommands once you are in the Kerberos Administration program. Commands which examine or modify the Kerberos database require that you enter your admin instance password at the `Admin password` prompt.

add_new_key *principal_name*

ank *principal_name*
Registers *principal_name* with the Kerberos database.

change_admin_password

cap
Changes your `admin` instance password.

change_password *principal_name*

cpw *principal_name*
Changes the Kerberos password for *principal_name*.

exit
Ends the KADMIN session with the Kerberos database. Alternatively, you can use the quit subcommand.

get_entry *principal_name*

get *principal_name*
Displays the entry for *principal_name* from the Kerberos database.

HELP *command name*
Displays help messages for KADMIN. If you enter this subcommand without an argument, a general help message is displayed.

list_requests

lr

? Displays a list of possible subcommands.

quit
Ends the KADMIN session with the Kerberos database. Alternatively, you can use the exit subcommand.

Context

- “Kerberos Remote Administrator Authorization Files” on page 360
- “KDB_EDIT Command” on page 369

KDB_DEST Command

Use the KDB_DEST command to erase the PRINCPL DAT and PRINCPL IDX files.

```
►►──KDB_DEST──────────────────────────────────────────────────────►◄
```

Operands

The KDB_DEST command has no operands.

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.

```
►►──KDB_EDIT──────────────────────────────────────────────────────►◄
```

Operands

The KDP_EDIT command has no operands.

Examples

This example shows how to register a Kerberos user or service using the KDB_EDIT command. After issuing the KDB_EDIT command:

1. The system prompts you for the Kerberos master key.

   ```
   Opening database...

   Enter Kerberos master:password:<krb_pw>
   ```

   Enter the valid master key at the master key prompt.
2. The system prompts you for the principal name.

   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 asks 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) [ 2020-01-01 ] ? <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.

To enter a null character, just press the **ENTER** key.

The following message is displayed if the user or service is registered:

```
Edit O.K.
Principle name:  <username>
```

4. The system prompts you for the next entry.

   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.

Context

- "KDB_UTIL Command" on page 371

KDB_INIT Command

Use the KDB_INIT command to create and format the Kerberos database.

```
►►──KDB_INIT──────────────────────────────────────────────────────────────►◄
```

Operands

The KDP_INIT command has no operands.

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 KRB CONF file. The default is `YOUR_KRB.RE.ALM`.

Then the system will prompt you for the master key. At the prompt, enter the master key.

Usage Notes

KDB_INIT creates two database files:

- PRINCPL DAT
- PRINCPL IDX

KDB_UTIL Command

Use the KDB_UTIL to dump or load the Kerberos database, or to change the Kerberos database master password.

```
►►──KDB_UTIL──┬─dump───────────┬─filename.filetype──────────────────────►◄
              ├─load───────────┤
              └─new_master_key─┘
```

Operands

dump
: Dump the contents of the Kerberos database to *filename filetype A*. The master password is contained within the file, so this file must be kept private.

load
: The contents of an existing Kerberos database are replaced with the contents of *filename filetype* with the exception of the default entry which is saved from the existing database.

new_master_key
: Changes the master password associated with an existing Kerberos database dump file *filename filetype*.

filename.filetype
Specifies the name of the file into which the database is dumped or from which the database is loaded, or that is to have its associated master password changed.

Examples

The following example uses the KDB_UTIL command to examine the contents of a database containing a single principal, *alan*. The database was dumped to a file named MYKERB DB with the command:

```
KDB_UTIL dump mykerb.db
```

The file MYKERB DB contains the following five records:

```
alan * 255 1 1 0 d742e777 820b480b 200001010459 199605281730 * *
changepw kerberos 255 1 1 0 fe3ac96d e518e3d3 200001010459 199605281726 db_creation *
default * 255 1 1 0 0 0 200001010459 199605281726 db_creation *
krbtgt endicott.ibm.com 255 1 1 0 19cc03b dbeaf5db 200001010459 199605281726 db_creation *
K M 255 1 1 0 b84c136f 51d511c 200001010459 199605281726 db_creation *
```

Note: Each line is a database entry. The four lines that contain the string `db_creation` must not be deleted.

A modified version of the file may be reloaded with the command:

```
KDB_UTIL load mykerb.db
```

To change the master password associated with the file, use the command:

```
KDB_UTIL new_master_key mykerb.db
```

Usage Notes

- You can edit the dumped file using a system editor to delete entries from the database. Certain information on each record is encrypted, so you cannot add or change any records; you must use KDB_EDIT instead. After deleting records, the database may be reloaded using the KDB_UTIL load function.
- The master password specified when the Kerberos authentication or administration server is started must the same as the master password that is associated with the loaded database file.
- To change the master password of the current Kerberos database:
  1. Dump the database using KDB_UTIL dump
  2. Change the master password using KDB_UTIL new_master_key
  3. Reload the database using KDB_UTIL load
  4. Store the new master password using KSTASH
  5. Restart the Kerberos servers

KSTASH Command

Use the KSTASH command to create the ETC K file. This file is needed if you do not want to enter the Kerberos master password manually and do not want to place the master password in clear text in a Kerberos authentication or administration server profile exit.

```
►►──KSTASH──────────────────────────────────────────────────────────►◄
```

Operands

The KSTASH command has no operands.

Usage Notes

You will be prompted for the Kerberos master key. The master key is case-sensitive.

Chapter 14. Configuring the LPD Server

The line printer daemon (LPD) virtual machine serves client requests to print a file. To configure the LPD server virtual machine, you must perform the following steps:

LPD Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the LPD server.
3. Customize the LPD CONFIG file.

Dynamic Server Operation

The LPD server provides a VM Special Message (SMSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information see "SMSG Interface to the LPD Server" on page 385.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  LPSERVE   0
```

The LPD server requires port TCP 515 to be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
PORT
  515  TCP LPSERVE    ; LPD Server
```

Step 2: Update the DTCPARMS File

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

```
:Nick.LPSERVE
  :Parms.
  :ESM_Enable.
  :ESM_Validate.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

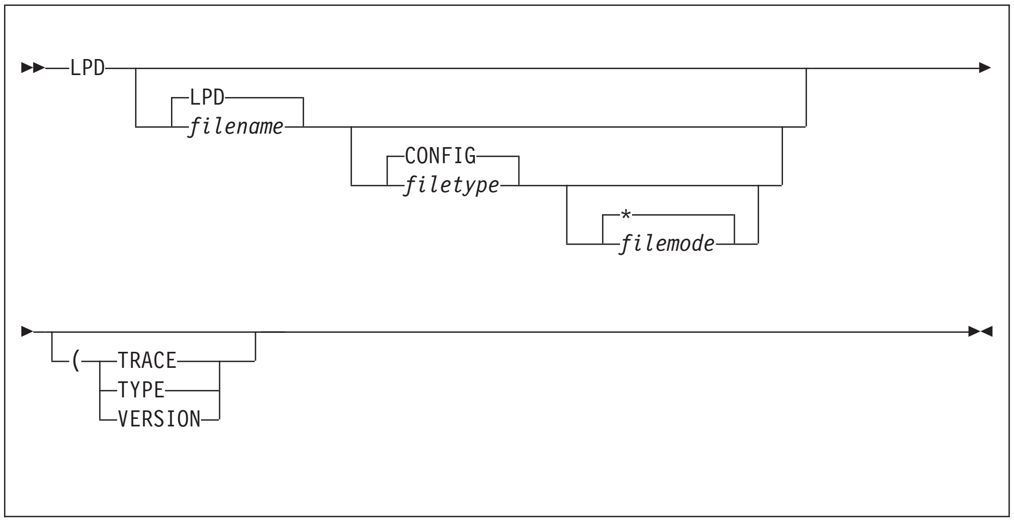
Note: You should modify the DTCPARMS file for the LPD server if you:

- Use a configuration file other than LPD CONFIG.

- Activate tracing using the TYPE and TRACE parameters.
- Want to display the version identifier.

LPD Command

LPD services are initiated using the LPD command:



Specify LPD command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

filename
The file name of the LPD server configuration file. The default file name is LPD.

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

filemode
The file mode of the configuration file. The default file mode is *.

TRACE
Causes a detailed trace of activities within the server to be recorded in the server console. Detailed tracing can also be activated by including the **DEBUG** statement in the LPD configuration file, or through use of the **TRACE ON** SMSG command.

TYPE
Causes a minimal trace of activities within the server to be recorded in the server console. Only significant events, such as the receipt of a job for printing, are recorded.

Note: The **TRACE OFF** SMSG command can be used to terminate event recording activated by the TRACE and TYPE operands.

VERSION
Causes LPD program version information to be written to the LPD server console.

If VERSION is the only operand specified, the LPD command terminates immediately after program version information is displayed.

Step 3: Customize the LPD CONFIG File

The LPD configuration file, LPD CONFIG, defines the services (printers and punches) supported and used by the LPD server. See “LPD Configuration File Statements” for detailed information about how to specify entries within this file. A sample configuration file is provided as LPD SCONFIG on the TCPMAINT 591 disk. Your customized LPD configuration file should be copied to the TCPMAINT 198 minidisk as LPD CONFIG.

Defining LP Services

Use the following statements to define LPD services. Each service description begins with a SERVICE statement that must be followed by either a LOCAL, REMOTE, or RSCS *type of service* statement.

Additional service attributes can be defined using the statements described in “Optional Service Statements” on page 381.

LPD Configuration File Statements

Specify LPD server parameters in the LPD configuration file as described in this section. Keep in mind the following when configuration statements are specified:

- Tokens are delimited by blanks and record boundaries.
- All characters to the right of, and including, a semicolon are treated as comments.
- **Case is significant** for the service, remote and filters statements.

DEBUG Statement

The DEBUG statement causes a detailed trace of activities within the server to be recorded in the server console.

```
►►──DEBUG──────────────────────────────────────────────────────────────►◄
```

The DEBUG statement has no operands.

Note: The **TRACE OFF** SMSG command can be used to terminate event recording activated by the DEBUG statement.

OBEY Statement

The OBEY statement identifies virtual machine user IDs that are authorized to use the SMSG interface to the LPD server.

```
        ┌──────────────┐
        ▼              │
►►──OBEY────user_id────┴──────────────────────────────────────►◄
```

Operands

user_id
: A user ID authorized to use the SMSG interface.

The number of user IDs that can be specified is limited only by the record length of the configuration file and the amount of virtual storage available in the LPD server virtual machine for construction of the applicable table. The OBEY statement may be repeated if necessary.

SMTP Statement

The SMTP statement identifies the SMTP server virtual machine defined for the local z/VM host. This statement is used in conjunction with the FAILEDJOB service statement. When an attempted print job fails and the FAILEDJOB MAIL statement is defined for a service, the LPSERVE virtual machine will generate a notice of the failure and send this notice to the user ID identified by the SMTP statement; this SMTP server then forwards this notice to the user that submitted the print request.

```
►►──SMTP──server_id─────────────────────────────────────────►◄
```

Operands

server_id
: The user ID of the SMTP virtual machine. If this statement is omitted, the default is SMTP.

SERVICE Statement

The SERVICE statement specifies a service for which connections are accepted and acknowledged.

```
►►──SERVICE──name──┬─PRINTER─┬──────────────────────────────►◄
                   ├─PUNCH───┤
                   └─NONE────┘
```

Operands

name
: The name of the service. The service *name* must be 1 to 8 characters in length. Only characters that are permitted in CMS file names are valid. **This value is case sensitive.**

PRINTER
Indicates that the service is a printer.

PUNCH
Indicates that the service is a punch device.

NONE
Not currently in use.

Type of Service Statements

The LOCAL, REMOTE, and RSCS statements described in this section are used to identify the manner in which files directed to the LPD server are processed. One of these *type of service* statements **must** be defined for each SERVICE statement that is included in the LPD server configuration file.

LOCAL Statement

The LOCAL statement specifies that files sent to the corresponding service are written to the local z/VM printer or punch.

```
           ┌─CLASS=A──────┐
►►──LOCAL──┼──────────────┼──┬────────────────┬─────────────────────────►◄
           └─CLASS=class──┘  ├─SPOOL=options──┤
                             └─TAG=tagtext────┘
```

Operands

CLASS=*class*
Defines a spool print class. Valid values are A-Z and 0-9.

TAG=*tagtext*
Defines information you want to associate with the specified spool file, such as *locid*, *userid*, and *priority* options. Do not include DEV or FILE operands as part of the *tagtext*. If specified, TAG=*tagtext* must be the last operand present for this configuration statement.

SPOOL=*options*
Defines operands for the CP SPOOL command. Any valid CP SPOOL command option may be specified. If specified, SPOOL=*options* must be the last operand present for this configuration statement.

Note: No case translation is performed on the values associated with the above operands. These operands and values are processed without modification, with respect to case. For more information on controlling LPD services from another operating system, see the *TCP/IP User's Guide*.

REMOTE Statement

The REMOTE statement specifies that files sent to the corresponding service are forwarded to a specified remote printer.

```
►►──REMOTE──printer@host──────────────────────────────────────────────►◄
```

Operands

printer@host
A destination printer at a specified internet node host.

The *printer* portion of this operand is case sensitive.

RSCS Statement

The RSCS statement specifies that files sent to the corresponding service are delivered to RSCS.

```
        ┌─CLASS=A─────┐
►►──RSCS─┼─────────────┼────────────────────────────────────────────────►
        └─CLASS=class─┘

  ┌─DEST=localid───IDENTIFIER=SYSTEM──PRIORITY=50─────────────────┐
►─┼────────────────────────────────────────────────────────────────┼──────►◄
  ├─TAG=tagtext────────────────────────────────────────────────────┤
  │ ┌─DEST=localid─┐  ┌─IDENTIFIER=SYSTEM────┐   ┌─PRIORITY=50─┐   │
  └─┼──────────────┼──┼──────────────────────┼───┼─────────────┼───┘
    └─DEST=node────┘  └─IDENTIFIER=identinfo─┘   └─PRIORITY=nn─┘ ┌─SPOOL=options─────┐
                                                                 └─OTHERS=tagoptions─┘
```

Operands

CLASS=*class*
Defines a spool print class. Valid values are A-Z and 0-9.

DEST=*node*
Defines the NJE (RSCS) node name assigned to a printer, punch or host. If no value is specified, the default is the local system.

TAG=*tagtext*
Defines information you want to associate with the specified spool file, such as *locid*, *userid*, and *priority* options. Do not include DEv or FIle operands as part of the *tagtext*. If specified, TAG=*tagtext* must be the last operand present for this configuration statement.

IDENTIFIER=SYSTEM
When *node* represents a host, SYSTEM indicates the supplied file is to be printed or punched using any available device. Otherwise, the file is directed to the specific printer or punch.

IDENTIFIER=*identinfo*
Provides information about where (on the destination node) the supplied file is to be directed. Identifier information *identinfo* can be specified using one of the following:

JOB
indicates the supplied file is to be submitted to host *node* for subsequent interpretation and execution. JOB is usually used to submit JCL to an MVS or VSE host.

userid
Indicates the supplied file is to be sent to the specified user ID (*userid*) at host *node*.

linkid
indicates the supplied file is to be directed to the specified link (*linkid*) at remote host *node*. Use this form when the specified link name is not known to the local host.

PRIORITY=*nn*
Defines a specific RSCS transmission priority where *nn* is a decimal number in the range 0-99. This operand is applicable only to RSCS links that use priority-based queuing.

SPOOL=*options*
Defines operands for the CP SPOOL command. Any valid CP SPOOL command option may be specified. If specified, SPOOL=*options* must be the last operand present for this configuration statement.

OTHERS=*tagoptions*
Defines any RSCS link, system, or user option recognized for the link type. If specified, OTHERS=*tagoptions* must be the last operand present for this configuration statement.

Note: No case translation is performed on the values associated with the above operands. These operands and values are processed without modification, with respect to case. For more information on controlling LPD services from other systems, see the *TCP/IP User's Guide*.

Optional Service Statements

Use the following statements to define general LPD service attributes.

EXIT Statement

The EXIT statement identifies an exit EXEC to run after spooling and tagging operations but before a virtual device is detached.

Two parameters are passed to the specified EXEC:
1. The virtual address of the spooled unit record device.
2. The three digit job number assigned to the print job submitted by the LPR client. The job number is used as part of the control and data files associated with the print job.

```
►►──EXIT───┬─START─┬──filename──────────────────────────────────────►◄
           └─END───┘
```

Operands

START
Indicates the exit EXEC is to be invoked after spooling and tagging, but before closing processing.

END
Indicates that the exit EXEC is to be invoked after closing processing but before the virtual device is detached.

filename
The name of the exit EXEC to be invoked.

FAILEDJOB Statement

The FAILEDJOB statement controls error notification handling for failed jobs (print requests). When a job fails, an error notification can be sent to the submitter of the print request, or the failure can be handled without notice. The failed job is discarded regardless of whether a notification is sent.

```
►►──FAILEDJOB──┬─MAIL────┬──────────────────────────────────────────────►◄
               └─DISCARD─┘
```

Operands

MAIL
Causes an error notification to be sent (using SMTP services) to the submitter of the print job.

DISCARD
Causes failed jobs to be discarded without notice.

Usage Notes

- When the FAILEDJOB statement is used with the LOCAL or RSCS statements, the default operand in effect is MAIL.
- When the FAILEDJOB statement is used with the REMOTE statement, the default operand in effect is DISCARD.

FILTERS Statement

The FILTERS statement can be used to control the type of printing or punching allowed for a specific LPD service. **Filter operands are case sensitive.**

```
►►──FILTERS──┬───┬──┬───┬──┬───┬──┬───┬──┬──────┬───────────────────────►◄
             └─f─┘  └─l─┘  └─p─┘  └─r─┘  └─lASf─┘
```

Operands

f Paginates a file using a page size, and truncates lines that exceed a specified maximum length.

l Prints a file leaving control characters intact.

p Paginates a file and adds a title, date, and page numbers.

r Prints a file, interpreting the first character of each line as FORTRAN (ASA) carriage control.

lASf
: Forces an LPD service to treat all files that include the **l** filter specification to use the **f** filter instead. Case is significant for this operand. Use of this option implicitly defines the **l** and **f** filters as being supported by the corresponding service.

Most printer services can accommodate all, or a combination of, the FILTERS statement operands. However, you may want to specify only **l** for punch devices.

LINESIZE Statement

The LINESIZE statement specifies the line length to be applied when a filter is used to paginate a file. This statement only applies to services that are designated as either LOCAL or RSCS.

```
   ┌─LINESIZE 132────┐
►►─┴─LINESIZE length─┴───────────────────────────────────────────────────►◄
```

Operands

length
: The character length of lines on a page; lines greater than this length are truncated. The default is 132. LINESIZE *length* values that fall within a certain range of values result in various virtual printer devices being defined for print operations, as follows:

Length Range	Printer Device
0-132	1403
133-150	3211
151-204	3800
205+	VAFP

PAGESIZE Statement

The PAGESIZE statement specifies the page length to be applied when a filer is used to paginate a file. This statement applies only to services that are designated as either LOCAL or RSCS.

```
               ┌─60────┐
►►──PAGESIZE───┴─lines─┴─────────────────────────────────────────────────►◄
```

Operands

lines
: The number of lines on a page. The default is 60.

RACF Statement

The RACF statement requires that users of the designated service provide their VM user ID and password as job description information supplied as part of their print request. It is *not* necessary to use an external security manager such as RACF with this option. See “Appendix A. Using TCP/IP with an External Security Manager” on page 617 for more information on using LPD with an external security manager.

```
►►──RACF──────────────────────────────────────────────────────────►◄
```

The RACF statement has no operands.

Note: Job description information is provided by the **JOB** option of the VM LPR command or the **-J** option of the Unix **lpr** command. For more information, see the section about controlling LPD services from other systems in the *TCP/IP User’s Guide*.

TRANSLATETABLE Statement

Specifies a translation table to be used for a specific print service. XLATETABLE is accepted as a synonym for this statement.

```
►►──TRANSLATEtable──tablename─────────────────────────────────────►◄
```

Operands

tablename
The file name of a translation table file to be used for EBCDIC to ASCII data translations; the file type for this file must be TCPXLBIN.

Note: If this statement is not specified, an attempt will be made to use the default translation table.

See “Chapter 29. Using Translation Tables” on page 595 for more information on translation tables.

Dynamic Server Operation

The VM Special Message Facility (SMSG) command provides an interface to the LPD server to:

- have CP commands executed while the server remains in operation.
- obtain information about jobs waiting to be printed.
- activate or deactivate server tracing.

Notes:

1. Responses to commands are **not** sent back to the originator of an SMSG command sent to the LPD server.

2. SMSG commands are accepted only from users specified in the OBEY statement of the LPD configuration file.

SMSG Interface to the LPD Server

```
►►──SMSG──server_id──┬─CP──command──────┬──────────────────────────────────►◄
                     ├─PRINT WORK───────┤
                     └─TRACE───┬─ON──┬──┘
                               └─OFF─┘
```

Operands

server_id
   The user ID of the LPD server virtual machine.

CP *command*
   Causes the specified *command* to be issued using a DIAGnose X'08' interface. If the command is not valid or fails, a record of the operating system response and the failing return code is written to the server console.

PRINT WORK
   Displays the current list of jobs waiting to be printed to the console. The list is prefaced by the string:

```
Work Queue start
```

   and terminated by the string:

```
Work Queue end
```

TRACE ON
   Activates full tracing of the processing within the LPD server virtual machine. Trace output is written to the server console.

TRACE OFF
   Deactivates tracing. This command terminates all tracing, regardless of the means used to establish tracing. This command also turns off the minimal tracing activated by the TYPE option of the LPD command.

Chapter 15. Configuring the RSCS Print Server

Instead of LPSERVE, the RSCS server may be chosen to provide an enhanced level of TCP/IP print support, including LPR and LPD.

Using RSCS provides end users with the following:

- Deferred (asynchronous) printing. The user's CMS session is not left idle waiting to print a file.
- TN3270E printer sessions, providing workstation print capabilities when LPD is not installed or available on the workstation.
- Enhanced error recovery. RSCS will periodically attempt to retransmit a print file in the event of a TCP/IP network or remote printer failure.
- Enhanced QUERY capabilities. RSCS commands are available to the end user to determine the status of their print request, complementing the existing LPQ and LPRM commands.
- The ability for workstation users to use LPR to print on any printer owned by the VM system. When an RSCS Version 3 Release 2 license is acquired, this capability is extended to any printer or user anywhere in the RSCS network.

In addition to the advantages provided to end users, it enables the integration of TCP/IP printing into an existing RSCS printer network and its associated management.

Configuring a TN3270E Printer

The TN3270E protocol supported by TCP/IP provides for the creation of 3270 printer sessions in addition to traditional display sessions. To make this capability available, the following steps must be performed:

1. An arbitrary name, called the ″LU name″ must be defined in the TN3270E control statement in the PROFILE TCPIP file. This statement assigns a virtual line address to the printer session.
2. The TN3270E Printer Management exit must be enabled by the TN3270EEXIT parameter of the INTERNALCLIENTPARMS statement in the PROFILE TCPIP file. See "INTERNALCLIENTPARMS Statement" on page 105.
3. An RSCS TN3270E link must be defined that has a line address that matches the line address defined in step 1. While not required, you may find administration and problem determination easier if the LU name is the same as the associated RSCS link name. The sample TN3270E Printer Management exit assumes the LU name and RSCS link names match.
4. The user must have the ″LU name″ configured in their TN3270E-capable emulator.

Configuring an RSCS LPR Link

To configure an RSCS LPR link, you will need to add LINKDEFINE and PARM configuration file statements in the RSCSTCP CONFIG file. Different options must be used depending on whether the printer is postscript or non-postscript.

RSCSTCP CONFIG Configuration File

The RSCSTCP CONFIG configuration file contains statements you can use to define your RSCS network. This file is read during initialization of the RSCS virtual machine. If this file is not found, RSCS initialization will fail. This file is the main configuration file for the RSCS server and will be stored on the TCP/IP Customization minidisk, TCPMAINT 198. It allows you to specify:

- The name of your local RSCS node if you wish it to be different from the system network ID
- LPR-type links for use with non-postscript printers
- LPR-type links for use with postscript printers
- LPD-type links for use as a local daemon
- TN3270E-type links for use with telnet attached 3287–1 printers
- UFT-type link for use as a local asynchronous UFT client

Two LPR links have been defined in the RSCSTCP CONFIG sample RSCS configuration file on TCPMAINT's 198 minidisk with the following *linkid*:

- LPR, which is to be used with non-postscript printers
- LPRP, which is to be used with postscript printers

Configuring a Non-Postscript Printer

This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an LPR link for use with non-postscript printers.

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

```
►►──LINKDEFine──LPR──AST──FOrm *──TYPE LPR─────────────────────────────►◄
```

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

The LPR keyword is in the *linkid* position. The *linkid* is a 1- to 8-character name of an LPR link that connects your local RSCS server to a remote line printer daemon (LPD) in a TCP/IP network. The LINK DEFINE statement must come before the PARM statement.

```
                                             ┌─TCPID=TCPIP─┐
►►──PARM──LPR──EXIT=LPRXONE──EParm='value'──ITO=0─┴─TCPID=tcpip─┴──────────────►

                      ┌─PORT=515────┐
►──┬────────────────┬─┴─PORT=portid─┴─USer=Yes──┬────────────────┬─────────►◄
   └─HOSTDafn=name──┘                           └─HOSTName=name──┘
```

Parameter	Description
EParm='*value*'	*value* is a character string up to 239 bytes in length and enclosed in single quotation marks (' '). For additional information see "Available EPARMs for Non-Postscript Printers".
PORT=*portid*	Specifies the port number to use when connecting to a remote host. The default is **well known** port 515.
TCPID=*tcpip*	Specifies the name of the TCP/IP server; the default name is TCPIP.
HOSTDafn=*name*	Specifies a 1- to 8-character name which should be used as the host name portion of the control and data file names. A control or data file name is used within the 'Receive control file' and 'Receive data file' subcommands of the daemon 'Receive job' command. The name should start with "cfa" (control file) or "dfa" (data file), followed by a three-digit job number, followed by the host name that has constructed the control/data file, as described in RFC 1179. RSCS defaults to using the link name for the host name portion of the file name if this parameter is not specified. Some daemons will not accept any name, other than the host name. This parameter allows you to specify the host name.
HOSTName=*name*	Specifies a 1- to 200-character fully qualified name of the remote host.

Available EPARMs for Non-Postscript Printers

The RSCS LPRXONE exit routine performs function for a non-postscript printer. It also performs simple translation of data to ASCII. The following EPARM values can be used to configure the LPRXONE exit behavior. Note that because the EPARM parameter is limited to 239 bytes of data, these options may be useful for only small amounts of data.

Parameter	Description
FF=	Determines how printer form-feeds are performed. **TOP** Form-feed is sent in the front of the file **Bottom** Form-feed is sent after the file; this is the default. **None** A form-feed is not sent.
FIlter=*f*	Specifies the printer filter used in the control file sent to the daemon if a filter is not specified by the user when the LPR command is issued; the default is f. One EBCDIC character is passed. If it is uppercase alphabetic, it will be translated to lowercase. No other validation is performed. This maintains consistency with RFC1179.
Prefix=*hex_string*	Optionally specifies a hexadecimal string to be sent in front of each file if a prefix string is not passed by the user when the LPR

command is issued; this string is not translated. This string can be used to pass printer specific setup values. Up to 200 bytes of data can be specified. By default, a prefix string is not sent with each file.

Sep=
Indicates if a separator page will be printed for each file.

- **Yes** Prints a separator page; this is the default. The origin user ID and node ID and distribution information are printed in large characters; other file information is printed in small characters in the Times-Bold font.
- **No** Does not print a separator page
- **Host** Sends the L control file record to request that the remote host produce the separator page.

SUFfix= *hex_string*
Optionally specifies a hexadecimal string to be sent after each file if a suffix string is not passed by the user when the LPR command is issued; this string is not translated. This string can be used to reset the printer upon print completion. Up to 200 bytes of data can be specified. By default, a suffix string is not sent with each file.

Config=LPR
Causes the LPRXONE exit to read the RSCSLPR CONFIG sample configuration file located on TCPMAINT's 198 minidisk. The configuration file can contain translation tables to override the tables used within the exit.

An * in column one denotes a comment line. Any line that does not have an * in column one will be interpreted as a configuration entry. All entries must be capitalized.

The following configuration records are supported:

ASCII=
Provide a table for translating ASCII control characters, overriding the default used by the exit. LPRXONE uses this translation table when files are already in ASCII and the user ID field of the TAG is set to 'ASCIIC'. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **ASCII=** records to replace the 256-byte translation table.

DOMAINAME=
Specifies a domain name, up to 255 characters, to be appended after the host name of the 'H' control file record. A period (.) will be inserted between the host name and domain name. This record can be used to add a domain name after the host name, which by default is the node name where the file originated or as specified in the **HOSTNAME=** record.

HOSTNAME=
Used to specify a host name, up to 255 characters, for the 'H' control file record, overriding the default, which is the node name where the file originated.

TOASCII=
Provide a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512

hexadecimal (0-9, A-F) characters may be specified on multiple **TOASCII=** records to replace the 256-byte translation table.

TOASCIIC= Provide a table for EBCDIC to ASCII translation of the LPR control file, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **TOASCIIC=** records to replace the 256-byte translation table.

USERNAME= Specifies a user name, up to 32 characters, for the ’P’ control file record, overriding the default name used by the exit, which is the user name of the file originator. This record can be used to cause all error messages to be sent to a central location.

A sample EPARM follows:

```
EPARM=’C=LPR S=N FF=N’
```

Configuring a Postscript Printer

This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an LPR link for use with postscript printers.

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

```
►►──LINKDEFine──LPRP──AST──FOrm *──TYPE LPR──────────────────────────────►◄
```

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

The LPRP keyword is in the *linkid* position. The *linkid* is a 1- to 8-character name of an LPR link that connects your local RSCS server to a remote line printer daemon (LPD) in a TCP/IP network. The LINK DEFINE statement must come before the PARM statement.

```
                                                 ┌─TCPID=TCPIP─┐
►►──PARM──LPRP──EXIT=LPRXPSE──EParm='value'──ITO=0─┼─────────────┼──────────►
                                                 └─TCPID=tcpip─┘

                     ┌─PORT=515────┐
►─┬────────────────┬─┼─────────────┼─USer=Yes─┬─────────────────┬──────────►◄
  └─HOSTDafn=name──┘ └─PORT=portid─┘          └─HOSTName=name───┘
```

Parameter	Description
EParm='*value*'	*value* is a character string up to 239 bytes in length and enclosed in

single quotation marks (' '). For additional information see "Available EPARMs for Postscript Printers".

PORT=*portid* Specifies the port number to use when connecting to a remote host. The default is **well known** port 515

TCPID=*tcpip* Specifies the name of the TCP/IP server; the default name is TCPIP.

HOSTDafn=*name*
Specifies a 1- to 8-character name that should be used as the host name portion of the control and data file names. A control or data file name is used within the 'Receive control file' and 'Receive data file' subcommands of the daemon 'Receive job' command. The name should start with "cfa" (control file) or "dfa" (data file), followed by a three-digit job number, followed by the host name that has constructed the control/data file, as described in RFC 1179. RSCS defaults to using the link name for the host name portion of the file name if this parameter is not specified. Some daemons will not accept any name, other than the host name. This parameter allows you to specify the host name.

HOSTName=*name*
Specifies a 1-to 200-character fully qualified name of the remote host.

Available EPARMs for Postscript Printers

The RSCS LPRXPSE exit routine performs functions for a postscript printer. This exit routine assumes that the remote printer is PostScript only, or that it will switch into PostScript mode when it receives a "%!PS" string following an EOT character. It also performs simple translation of data to ASCII. The following EPARM values can be used to configure the LPRXPSE exit behavior. Note that because the EPARM parameter is limited to 239 bytes of data, these options may be useful for only small amounts of data.

Ehandler= Determines if a PostScript error handler will be downloaded to the printer the first time a file is sent to the printer after the link is started. This error handler enables any errors to be printed, so the information will not be lost.
Yes The error handler is downloaded; this is the default.
No The error handler is not downloaded.

EOT=

Yes EOT characters will be inserted after the separator page, data file, and trailer page; this is the default.

No EOT characters will not be inserted.

FIlter=*f* Specifies the printer filter used in the control file sent to the daemon if a filter is not specified by the user when the LPR command is issued; the default is f. One EBCDIC character is passed. If it is uppercase alphabetic, it will be translated to lowercase. No other validation is performed. This maintains consistency with RFC1179.

FOrm=*OrFnFsLs*
Specifies the default form to use when printing plain text files and when a form has not been specified on the LPR command.

The following can be used for *OrFnFsLs* and shows the defaults.

Or File orientation:

PO	Portrait (default)
LA	Landscape

Fn Font name code

CB	Courier-Bold
CI	Courier-Oblique
CP	Courier (default)
CX	Courier-BoldOblique
HB	Helvetica-Bold
HP	Helvetica
HI	Helvetica-Oblique
HX	Helvetica-BoldOblique
SP	Symbol
TB	Times-Bold
TI	Times-Italic
TP	Times-Roman
TX	Times-BoldItalic

Fs Font size, 04 thru 99; the default is 11 for portrait and 10 for landscape orientation

Ls Additional leading size, 0.0 - 9.9, added to font size to give leading; the default is 09 for portrait and 12 for landscape

Prefix= *hex_string*
Optionally specifies a hexadecimal string to be sent in front of each file if a prefix string is not passed by the user when the LPR command is issued; this string is not translated. This string can be used to pass printer specific setup values. Up to 200 bytes of data can be specified. By default, a prefix string is not sent with each file.

Sep= Indicates if a separator page will be printed for each file.

Yes
Prints a separator page; this is the default. The origin user ID and node ID and distribution information are printed in large characters; other file information is printed in small characters in the Times-Bold font.

No
Does not print a separator page

Host
Sends the L control file record to request that the remote host produce the separator page.

SUFfix= *hex_string*
Optionally specifies a hexadecimal string to be sent after each file if a suffix string is not passed by the user when the LPR command is issued; this string is not translated. This string can be used to reset the printer upon print completion. Up to 200 bytes of data can be specified. By default, a suffix string is not sent with each file.

Trailer= Indicates if a trailer page will be printed

Yes
Prints a trailer page after the file. It is identical to the header page with the addition of a count of the bytes in the file.

No
Trailer page is not printed; this is the default.

Config=LPRP
Causes the LPRXPSE exit to read the RSCSLPRP CONFIG sample configuration file located on TCPMAINT's 198 minidisk.

This configuration file is used to supply the following:

- To override the default translate table.
- To override the postscript program sent to the printer when printing plain text files.
- Additional font names used when printing plain text files.

An * in column one denotes a comment line. Any line that does not have an * in column one will be interpreted as a configuration entry. All entries must be capitalized.

The following configuration records are supported:

DOMAINNAME=
Specifies a domain name, up to 255 characters, to be appended after the host name of the 'H' control file record. A period (.) will be inserted between the host name and domain name. This record can be used to add a domain name after the host name, which by default is the node name where the file originated or as specified in the **HOSTNAME=** record.

HOSTNAME=
Used to specify a host name, up to 255 characters, for the 'H' control file record, overriding the default, which is the node name where the file originated.

FONT=
Provides a 2-character font abbreviation followed by a 32-character font name. There should be no blanks between the abbreviation and full font name. Multiple records can be provided for supplying as many additional fonts as required. The abbreviation should be unique on each **FONT=** record. The fonts must be installed on the printer. The current available font names are:

2-Character Abbreviation	32-Character Font Name
CB	Courier-Bold
CI	Courier-Oblique
CP	Courier (exit default)
CX	Courier-BoldOblique
HB	Helvetica-Bold
HP	Helvetica
HI	Helvetica-Oblique
HX	Helvetica-BoldOblique
SP	Symbol
TB	Times-Bold
TI	Times-Italic
TP	Times-Roman

TX	Times-BoldItalic

PSCRIPT=
Provide a replacement postscript program to be used when printing a plain text file. The postscript program must be enclosed within quotes. Anything after the ending quote will be ignored allowing for comments.

For example:

```
PSCRIPT='this is line one'  comment for line one
PSCRIPT='this is line two'
```

Multiple **PSCRIPT=** records can be provided in order to supply the entire program. LPRXPSE will add a carriage return (X'0A') after each record, and will translate the record from EBCDIC to ASCII.

Note: When replacing the postscript program, the ability to tailor the file orientation, font name, font size, and additional leading size through a FORM is lost. The supplied postscript program must define all of these attributes.

TOASCII=
Provide a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **TOASCII=** records to replace the 256-byte translation table.

TOASCIIC=
Provide a table for EBCDIC to ASCII translation of the LPR control file, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **TOASCIIC=** records to replace the 256-byte translation table.

USERNAME=
Specifies a user name, up to 32 characters, for the 'P' control file record, overriding the default name used by the exit, which is the user name of the file originator. This record can be used to cause all error messages to be sent to a central location.

The following is a sample EPARM:

```
EPARM='C=LPRP S=N T=N'
```

Form Parameter of LPR Command When Printing to PostScript

When printing to postscript printers, you can also specify the following values on the FORM=*OrFnFsLs* operand of the LPR command.

Or
File orientation:

PO	Portrait
LA	Landscape

Fn
Font name code

CB	Courier-Bold
CI	Courier-Oblique
CP	Courier
CX	Courier-BoldOblique
HB	Helvetica-Bold
HP	Helvetica

HI	Helvetica-Oblique
HX	Helvetica-BoldOblique
SP	Symbol
TB	Times-Bold
TI	Times-Italic
TP	Times-Roman
TX	Times-BoldItalic

Fs Font size, 04 thru 99.

Ls Additional leading size, 0.0 - 9.9, added to font size to give leading.

Configuring an RSCS LPD Link

The RSCSLPD CONFIG configuration file contains statements you can use to customize an RSCS LPD-type link. This file is read during link initialization. If this file is not found, the link initialization will fail. This file will be stored on the TCP/IP Customization minidisk, TCPMAINT 198. It allows you to specify:

- A replacement ASCII to EBCDIC translation table
- A replacement EBCDIC to ASCII translation table
- Definitions to be used while receiving a file for a particular printer queue name. The definitions which may be defined are
  - a queue name
  - logical record length
  - number of lines per page
  - spool file class
  - spool file form
  - job name
  - PSF destination
  - whether or not to paginate
  - whether or not to translate
  - destination user ID
  - destination node ID
  - TCPXLBIN translation table to use for ASCII to EBCDIC translations

To configure an RSCS LPD link, you will need to add LINKDEFINE and PARM configuration file statements in the RSCSTCP CONFIG file. One LPD link has been defined in the RSCSTCP CONFIG sample RSCS configuration file on TCPMAINT's 198 minidisk.

This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an LPD link.

```
►►──LINKDEFine──LPD──TYPE LPD──────────────────────────────────────────►◄
```

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

The LPD keyword is in the *linkid* position. The *linkid* is a 1- to 8-character name of an LPD link that will act as a gateway accepting print data streams from the TCP/IP environment.

```
                                                 ┌─TCPID=TCPIP─┐
►►──PARM──LPD──EXIT=LPDXMANY──┬────────────────┬─┼─────────────┼────────────────────►
                              └─EParm='value'──┘ └─TCPID=tcpip─┘

  ┌─PORT=515────┐ ┌─Timeout=60──┐
►─┼─────────────┼─┼─────────────┼──────────────────────────────────────────────────►◄
  └─PORT=portid─┘ └─Timeout=nnn─┘
```

Parameter	Description
EParm='*value*'	*value* is a character string up to 239 bytes in length and enclosed in single quotation marks (' '). For additional information see "Available EPARMs for LPD Links".
TCPID=*tcpip*	Specifies the name of the TCPIP server; The default name is TCPIP.
PORT=*portid*	Specifies the port number to use when listening on the local host. The default is **well known** port 515.
Timeout=*nnn*	Specifies the amount of time in seconds RSCS will wait for data when receiving from a TCP/IP line print router prior to breaking the socket connection; the default is 60. After the LPD link driver breaks this connection, it will remain operational waiting for another line print router to connect.

Available EPARMs for LPD Links

The RSCS LPDXMANY exit routine performs functions to support a single LPD printer queue. Multiple LPD link drivers can be defined using this exit. It also performs simple translation of data to EBCDIC.

The spool file created will be of type VAFP. A record width up to 1280 characters will be supported. Any records received with a width greater than 1280 characters will be split into multiple records. The CMS receive command will create a file with a record length of 204. Any data in a record past 204 will be truncated. CP will also truncate the data with a record length greater than 204 if the spool file is destined to a CP system printer.

Printer job commands received from the line print router contain a queue name of the form *userid@nodeid* or *userid%nodeid*. The following examples show how the LPDXMANY routine parses this queue name, and how the provided information affects a created spool file.

KERRY@MAINE	Will set the node ID to MAINE and the user ID to KERRY. The file will be sent to user KERRY at node MAINE.
KERRY%MAINE	Will set the node ID to MAINE and the user ID to KERRY. The file will be sent to user KERRY at node MAINE.
KERRY@	Will set the node ID to the local node name and the user ID to KERRY. The file will be spooled to user KERRY on the local system.

KERRY%	Will set the node ID to the local node name and the user ID to KERRY. The file will be spooled to user KERRY on the local system.
@LASER1	Will set the node ID to LASER1 and the user ID to SYSTEM causing the file to be sent to the network node LASER1.
%LASER2	Will set the node ID to LASER2 and the user ID to SYSTEM causing the file to be sent to the network node LASER2.
LASER3	Will set the node ID to LASER3 and the user ID to SYSTEM causing the file to be sent to the network node LASER3.

Note: The LPDXMANY routine will limit the length of the user ID and node ID fields to 8 characters. Any extra data in those operand fields will be discarded.

The following EPARM values can be used to configure the LPDXMANY exit behavior. Note that because the EPARM parameter is limited to 239 bytes of data, these options may be useful for only small amounts of data.

Parameter **Description**

Config=*LPD*
: Causes the LPDXMANY exit to read the RSCSLPD CONFIG sample configuration file located on TCPMAINT's 198 minidisk. This configuration file is used to supply the following:

- Translation table to override the one used by the exit.
- Supply overrides for processing when a file is received from a remote LPR command based on the printer queue name.

An * in column one denotes a comment line. Any line that does not have an * in column one will be interpreted as a configuration entry. All entries must be capitalized.

The following configuration records are supported:

TOASCII=
: Provide a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **TOASCII=** records to replace the 256-byte translation table.

TOEBCDIC=
: Provide a table for ASCII to EBCDIC translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **TOEBCDIC=** records to replace the 256-byte translation table.

queuename
: Provide the ability to override defaults used by LPDXMANY on a printer queue name basis when receiving a file from a remote host. Multiple, unique, printer queue name records can be specified.

Format of Printer Queue Name Records

The layout of the printer queue name records is as follows:

- Each queue name option is separated by one or more blanks.
- The parameters are not column dependent.
- One record per line, continuation is not supported.
- Each parameter IS position dependent.

- An * can be used in a position (other than for the printer queue name) to tell LPDXMANY to use the existing default.
- The printer queue name will be parsed into user ID and node ID as follows:
  <userid@>nodeid
  <userid%>nodeid

nodeid	user ID will be set to SYSTEM.
@nodeid	user ID will be set to SYSTEM.
%nodeid	user ID will be set to SYSTEM.
userid@	node ID will be set to local node name.
userid%	node ID will be set to local node name.
NONEOFTHEABOVE	node ID and user ID MUST be set within record.

  The user ID and node ID parsed from the printer queue name can be overridden within the record. If both are overridden, the printer queue name is not parsed. If neither are overridden, the printer queue name is parsed into user ID and node ID which must be valid (either of which can still be overridden within).

```
queuename  ppos lpage class forms jobn dest pagination translation userid nodeid tcpxlbin
DEFAULT ppos lpage class forms jobn dest pagination translation userid nodeid tcpxlbin
```

Parameter	Description
queuename	is a printer queue name up to 32 characters. A *queuename* of DEFAULT can be used to define parameters for any printer queue not defined by a configuration file record. When a printer queue name arrives, LPDXMANY first looks for a configuration record of that name. If this is not found, LPDXMANY will look for a configuration record using DEFAULT; if this is not found, it will use existing LPDXMANY defaults.
ppos	is the logical record length 1-1280; the default is 255.
lpage	is the number of lines per page 1-99; the default is 66.
class	is the 1-character spool file class; the default is blank.
forms	is the 1- to 8-character spool file form; the default is blank.
jobn	is the 1- to 8-character job name; the default is SYSTEM.
dest	is the 1- to 8-character PSF destination; the default is blank. When using a PSF destination, LPDXMANY will set the user ID field of the tag to *SYSTEM*.
pagination	is either PAGE or NOPAGE; the default is NOPAGE. This parameter determines how pagination is performed: PAGE will cause LPDXMANY to always paginate, regardless of the print filter. NOPAGE will cause LPDXMANY to paginate only as defined by the control file filters 'f' or 'p'. To be effective, the control file must be received prior to the data file to be printed.
translation	is either ASISCC, NOTRAN or TRAN; the default is TRAN. Translation determines if the data file is translated prior to spooling.
userid	is the 1-to 8-character user ID that will receive the spooled file.
nodeid	is the one- to eight-character destination

tcpxlbin	is the file name of a TCP/IP TCPXLBIN translate table to be used for this printer queue name.

QUEUENAME examples:

```
LPR@NODEONE 1280 50 * STDN * * * ASISCC ASCII
DEFAULT 255 66 * * SYSTEM * NOPAGE TRAN *
```

The first example would allow a file to be received without translation, and spooled to an LPR link. ASCII tells the LPRXONE exits that the file is already in ASCII and to send it unaltered to the remote daemon.

The second example would allow a file to be spooled to the system printer.

The following is a sample EPARM:

```
EPARM='C=LPD'
```

Configuring an RSCS TN3270E Printer Link

To configure an RSCS TN3270E printer link, you will need to add LINKDEFINE and PARM configuration file statements in the RSCSTCP CONFIG file. You must define one TN3270E printer link for each printer LUNAME defined within the TCPIP configuration file.

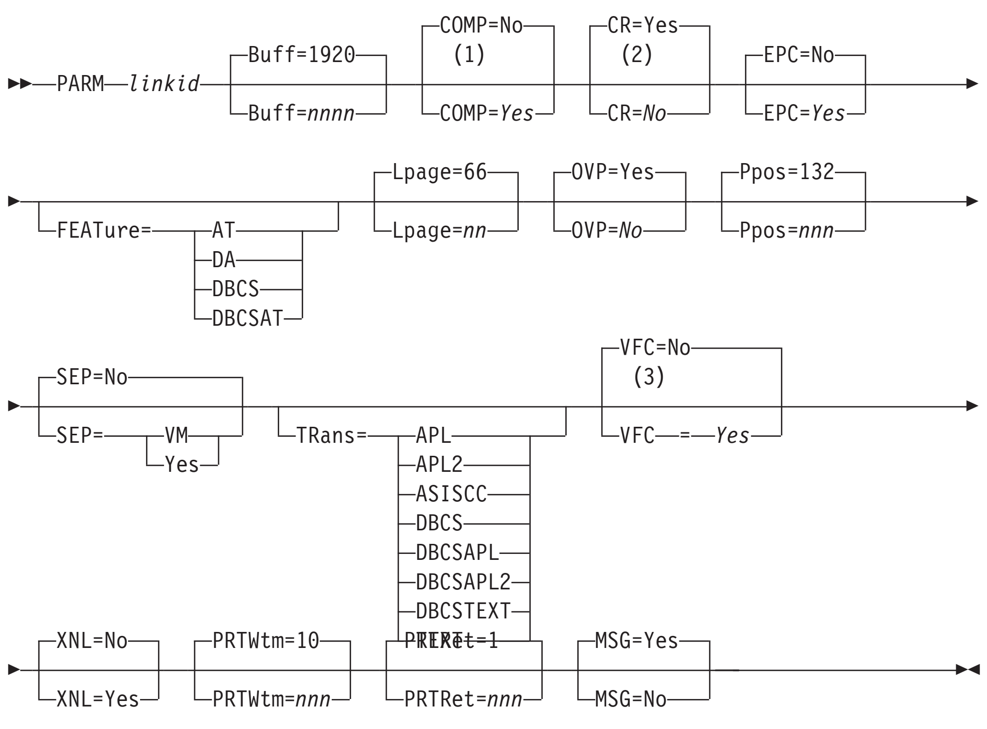
This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an TN3270E printer link. The *linkid* may match the LUNAME; however, the line address must match the virtual address defined in the TCPIP configuration file.

```
►►──LINKDEFine──linkid──AST──LINE── ──vaddr──TYPE── ──TN3270E────────────────►◄
```

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

You can specify multiple LINKDEFINE statements for the same link. If you specify more than one LINKDEFINE statement with the same *linkid*, RSCS uses the attribute from the last LINKDEFINE statement.



Notes:

1 Specify NO for locally attached printers.

2 CR has no meaning unless VFC=YES is also specified.

3 Generally it is recommended to specify VFC=YES.

linkid
: Is the 1- to 8-character name of the link that connects your local RSCS system to a TN3270E attached printer. Before you can specify a PARM statement for a link, you must specify a LINKDEFINE (or LINK) statement with the same link ID.

Buff=1920

Buff=*nnnn*
: Is the buffer size of the 3287–1 printer. The following buffer sizes are valid: 480, 960, 1920, 2560, 3440, and 3564. The correct buffer size has to be specified to match the buffer size on the printer being used. If this keyword is omitted, the buffer size defaults to 1920.

COMP=Yes

COMP=No
: Specifies that the link will perform blank compression. If omitted, the default is to not compress.

CR=Yes

CR=No
: Specifies if a carriage return (CR) should be sent after a forms feed (FF) when VFC=YES is specified. CR has no meaning unless VFC=YES is also specified. If omitted, the default is NO.

EPC=Yes

EPC=No
Specifies if Early Print Complete should be used by the line driver. If omitted, the default is NO.

FEATure=
Specifies that the printer has the following features:

AT
APL/Text feature

DA
Data Analysis-APL feature

DBCS
Double-Byte Character Set feature

DBCSAT
AT and DBCS features

Lpage=66

Lpage=*nn*
Specifies the number of lines per page on the type of form inserted in the 3270 printer. The value may be from 0 to 99. A value of 0 specifies that no page ejects will be done when a page eject is read in the file. If this keyword is omitted, the length of a page defaults to 66.

Note: This value is used by RSCS only to determine the current vertical position on a page while printing a file. Any actual page ejects are determined by Skip-to-Channel-1 CCWs within the file. If there are no Skip-to-Channel-1 CCWs in the file, no page ejects are done, except between files. The actual number of lines contained in the file between consecutive Skip-to-Channel-1 CCWs must be equal to or less than this value.

OVP=Yes

OVP=No
Specifies if the link will allow overprinting to occur (such as underscored words or overstruck characters for highlighting). If omitted, the default is to allow overprinting. Specify NO for those printers that do not support the 3270 CR (Carriage Return) order.

Ppos=132

Ppos=*nnn*
Specifies the maximum number of print positions available on the 3270 printer. This value may be between 1 and 220, depending on the actual printer. Consult the *Component Description* manual for the specific printer to determine this value. If omitted, the number of print positions defaults to 132.

PPOS is a hardware maximum determined by the number of characters the printer *can* print on each line, independent of the size of the form used in the printer. Each printer has its own PPOS value. If the PPOS value you specify does not match the printer's PPOS value, some types of printed output (including separator pages) will not print correctly.

SEP=
Specifies if a separator page will be inserted before each print file and sent across the link.

Yes
For RSCS-style separator pages.

No
For no separator pages. This is the default.

VM
: For VM-style separator pages.

TRans=
: Specifies the default translation mode for files being transmitted to a printer having the DA, AT, DBCS, or DBCSAT feature, or specifies that unprintable characters should not be translated into blanks (the ASISCC operand). Specifying TRANS tells RSCS that all files may contain the special characters you are indicating and tells RSCS to translate these special characters accordingly. However, the default translation mode you specify on TRANS can be overridden by using the PRT option of the CP TAG command.

APL
: If RSCS detects internally represented EBCDIC special APL characters in a file, RSCS will translate those characters into the appropriate two-byte I/O interface codes.

APL2®
: If RSCS detects internally represented EBCDIC special APL2 character in a file, RSCS will translate those characters into the appropriate two-byte I/O interface codes.

This mode is only valid when FEATURE is set to DBCSAT or AT. If TRANS=APL2 is specified with any other FEATURE setting, it is marked as an incorrect combination and an error message is created.

ASISCC
: RSCS will not translate unprintable characters into blanks. This lets the user pass SCS orders to the remote printer.

TRANS=ASISCC means no translation occurs as a default. If a file needs translation, then override this using the TAG command (assuming you used the correct setting for the FEATURE operand of the START command.)

If the FEATURE option is not specified, only ASISCC, GRAPH, GRAF or NOTR can be used on the TAG PRT command.

Note: TRANS=ASISCC can be specified without a FEATURE option.

DBCS
: RSCS will verify double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters for valid DBCS syntax before transmitting a file.

DBCSAPL
: RSCS will translate internally represented EBCDIC special APL characters into the appropriate two-byte I/O interface codes and will verify all double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters for valid DBCS syntax before transmitting a file.

DBCSAPL2
: All files containing internally represented EBCDIC special APL2 characters are translated to the appropriate two-byte I/O interface codes. This mode is only valid when FEATURE is set to DBCSAT. If TRANS=DBCSAPL2 is specified with any other FEATURE setting, it is marked as an incorrect combination and an error message is created.

DBCSTEXT
: RSCS will translate internally represented EBCDIC special TEXT characters

into the appropriate two-byte I/O interface codes and will verify double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters for valid DBCS syntax before transmitting a file.

Text
RSCS will translate internally represented EBCDIC special TEXT characters into the appropriate two-byte I/O interface codes.

VFC=No

VFC=Yes
Specifies that the printer has the vertical forms control feature. When YES is specified, the page length must be set manually on the printer and the LPAGE value must match that setting. If omitted, the default is no vertical forms control, and vertical spacing is achieved through multiple New Line orders. YES is valid only for those printers that also support the 3270 CR (Carriage Return) order.

Note: Generally, it is recommend you specify VFC=YES. If users send GDDM or IPDS™ graphic files to this printer, you *must* specify VFC=YES to ensure that the pages align correctly.

XNL=No

XNL=Yes
Specifies that the link is to include an extra NL (New Line) order after each line that is as long as the maximum number of print positions. Specify YES for those printers that do not generate an extra new line function for a maximum length line.

PRTWtm
The time, in seconds (1-600), between recovery attempts due to an unattached device. If not specified, the default is 10 seconds.

PRTRet
The number of hours (1-100) the link will attempt error recovery due to an unattached device before terminating. If not specified, the default is 1 hour.

MSG=Yes

MSG=No
Specifies if RSCS will allow the TN3270E-type link to dequeue messages destined for transmission to the printer.

The following notes explain the use of the PARM statement for a TN3270E Type link:

1. When you define a form name (using the FORM statement), you can define the following characteristics:
   - Separator page style
   - Line length
   - Page length

   If you specify a form name when starting a link, be aware that the form name characteristics always override any SEP, PPOS, and LPAGE specifications.
2. If you specify an incorrect combination of TRANS and FEATURE settings, the SCO receives error message 807E and the link is deactivated.

Table 24. Correct Combinations for TRANS and FEATURE Settings

TRans=	FEATure= DA	FEATure= AT	FEATure= DBCS	FEATure= DBCSAT
APL	X	X		X
APL2		X		X
TEXT	X	X		X
DBCS			X	X
DBCSAPL				X
DBCSAPL2				X
DBCSTEXT				X
ASISCC	X	X	X	X

Note: If APL, APL2, or TEXT is specified for the TRANS operand, and the FEATURE operand is set to DBCSAT, then there is no DBCS translation of the file, only APL, APL2, or TEXT. The DBCS translation will only occur with APL, APL2, or TEXT translation when the TRANS setting is DBCSAPL, DBCSAPL2, or DBCSTEXT.

Note: TRANS=ASISCC does not require the FEATURE operand to be specified.

TAG Command for a TN3270E printer

The options for a TAG command to a TN3270E printer are described in the following sections. RSCS processes all the options until it meets a left parenthesis or the end of the command.

Do not include an option more than once in the same TAG command. If you do, RSCS uses the **last** value specified. For example, if you specify the following PRT options on a TAG command to a TN3270E printer, the value of PRT will be DBCSTEXT:

```
tag dev ... prt=asiscc ... prt=dbcstext
```

The following is a syntax diagram of the TAG command:

```
►►──TAg──DEv──00E──nodeid──┬────────────────────────────────────────┬──────►◄
                           └─linkid──┬──────────────────────────────┤
                                     └─50──┬────────────────────────┤
                                           └─┤ TN3270E Link Options ├─┘
```

TN3270E Link Options:

```
├───┬─PRT=───┬─APL───────┬─┬─────────────────────────────────────────────────┤
    │        ├─APL2──────┤ │
    │        ├─ASISCC────┤ │
    │        ├─DBCS──────┤ │
    │        ├─DBCSAPL───┤ │
    │        ├─DBCSAPL2──┤ │
    │        ├─DBCSTEXT──┤ │
    │        ├─GRAF──────┤ │
    │        ├─GRAPH─────┤ │
    │        ├─NOTR──────┤ │
    │        └─TEXT──────┘ │
    └─PRMODE=──────SOSI2───┘
```

nodeid
: Specifies the destination node ID for output created by the virtual device. You must specify SYSTEM if you also specify DEST on the SPOOL command, to allow DEST to have effect.

linkid
: Identifies the virtual machine link ID, the directly attached workstation, or the directly attached TN3270E printer link ID at the destination node (*nodeid*) that is to receive the output generated by the virtual device.

  If the link ID is omitted or specified as SYSTEM, the file will be transferred to the system printer or punch. You must specify a link ID (or SYSTEM) if you also specify a priority. You must specify SYSTEM if you also specify DEST on the SPOOL command, to allow DEST to have effect.

PRT=
: Overrides any translation mode (TRANS) specification of APL, APL2*, TEXT, or DBCS on the RSCS START command for TN3270E-type links. If no APL-, TEXT-, or DBCS-related PARM operands were specified when the link was started, PRT is ignored, except for NOTR, GRAF, GRAPH, or ASISCC.

  If PRT is not specified, the START PARM or OPARM (if specified) TRANS setting is the default translation used for the file.

  Alternatively, PRT can be used to identify a GDDM (Graphical Data Display Manager) spool file.

  The PRT can have the following values

  APL
  : Specifies that internally represented EBCDIC special APL characters will be translated to the appropriate two-byte I/O interface codes.

  APL2
  : All files containing internally represented EBCDIC special APL2 characters are translated to the appropriate two-byte I/O interface codes. This mode is only valid when FEATURE is set to DBCSAT or AT. If PRT=APL2 is specified with any other FEATURE setting, it is marked as an incorrect combination and an error message is created.

ASISCC
RSCS will not translate unprintable characters in files into blanks. This lets the user pass 3270 orders to the remote printer.

DBCS
Specifies that the file contains Double-Byte Character Set (DBCS) data.

DBCSAPL
Specifies that all files containing internally represented EBCDIC special APL characters are to be translated to the appropriate two-byte I/O interface codes and all double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters will be verified for valid DBCS syntax before being transmitted.

DBCSAPL2
Specifies that all files containing internally represented EBCDIC special APL2 characters are to be translated to the appropriate two-byte I/O interface codes and all double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters will be verified for valid DBCS syntax before being transmitted.

This mode is only valid when FEATURE is set to DBCSAT. If PRT=DBCSAPL2 is specified with any other FEATURE setting, it is marked as an incorrect combination and an error message is created.

DBCSTEXT
Specifies that all files containing internally represented EBCDIC special text characters are to be translated to the appropriate two-byte I/O interface codes and all double-byte EBCDIC character strings delimited by SO/SI (shift-out/shift-in) characters will be verified for valid DBCS syntax before being transmitted.

GRAPH

GRAF
Means that the file contains GDDM (Graphical Data Display Manager) output records. For translation to occur the file must be in *punch* format.

NOTR
Means that the internally represented EBCDIC special APL, APL2, or TEXT characters are not to be translated to the two byte I/O interface codes for this file. Only 3270 translation will occur for the removal of 3270 control characters.

TEXT
Specifies that internally represented EBCDIC special text characters will be translated to the appropriate two-byte I/O interface codes.

PRMODE=
Specifies if a blank character should print before and after a DBCS character. PRMODE=SOSI2 for TN3270E will result in the SO and SI being replaced with an SA order for DBCS and a reset and will occupy no space in the output line. When PRMODE is not specified, then the SO, SI will occupy 1 position in an output line.

SOSI2
Does not print a blank before and after a DBCS character string.

Chapter 16. Configuring the NDB Servers

This chapter describes how to configure the Network Database (NDB) system, which consists of:

- The NDB Port Manager server
- One or more NDB agent servers
- The NDB client.

Note: TCP/IP for DOS, OS/2, AIX on the RISC System/6000, or Sun Microsystems must be installed before, or in conjunction with, the NDB client.

The NDB Port Manager, one NDB agent server, and NDB client source code are installed with TCP/IP.

To configure the NDB system for your installation, you must perform the following steps:

NBD System Configuration Steps

1. Define additional NDB agent servers. (Optional)
2. Update PROFILE TCPIP.
3. Update the DTCPARMS file for the NDB Port Manager and NDB agent servers.
4. Provide NDB agents access to DB2 Server for VSE & VM.

Step 1: Define Additional NDB Agent Servers (Optional)

To run NDB, your installation must have one NDB Port Manager virtual machine and at least one NDB agent server defined. The default TCP/IP installation environment includes one NDB Port Manager and one NDB agent, named **NDBPMGR** and **NDBSRV01**, respectively.

With NDB, each NDB client makes a one-to-one connection with an NDB agent for each unit of work (UOW) performed. Thus, having more NDB agents available allows more clients to simultaneously access the database. Up to 20 NDB agents can be defined for use with a given TCP/IP server.

For any additional NDB agents that you define, it is recommended that you:

- maintain the NDBSRV*nn* naming convention
- model your CP directory entries after that supplied for the NDBSRV01 virtual machine.

See "Chapter 3. General TCP/IP Server Configuration" on page 17 for more information about duplicating existing servers. If necessary, consult the *TCP/IP Feature for z/VM, Level 3A0 Program Directory* for specific DASD storage and user ID requirements that may be applicable to these virtual machines.

Step 2: Update PROFILE TCPIP

Include the various NDB server virtual machines in the AUTOLOG statement of the TCPIP configuration file. These servers are then started automatically when TCPIP is initialized. Add a separate line for the NDP Port Manager and one each for the NDB agents you employ. Verify that the following statements have been added to the PROFILE TCPIP file:

```
AUTOLOG
  NDBPMGR  0         ; NDB Port Manager
  NDBSRV01 0         ; NDB Agent 1
  NDBSRV02 0         ; NDB Agent 2
⋮
  NDBSRV20 0         ; NDB Agent 20
```

Note: The NDB Port Manager virtual machine must be logged on before any NDB agents.

Step 3: Update the DTCPARMS File

NDB Port Manager

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

```
:Nick.NDBPMGR
```

NDB Agent Servers

Up to 20 NDB agents (named NDBSRV01 through NDBSRV20, by convention) can be used with a given TCP/IP server. When these agents are started, the TCP/IP server initialization program searches specific DTCPARMS files for configuration definitions that apply to these servers. Tags that affect the NDB agent servers are:

```
:Nick.NDBSRVnn
  :ESM_Enable.
  :ESM_Validate.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the NDB servers if you:
- Use an ESM to authenticate and authorize access to resources managed by the server.
- Use the RPIVAL or another module for validating user IDs and passwords.

PORTSRVS Command

The PORTSRVS command is used to initialize the NDB Port Manager:

```
►►──PORTSRVS──────────────────────────────────────────────────────────►◄
```

The PORTSRVS command has no operands.

PORTCLNT Command

The PORTCLNT command is used to initialize the NDB agent servers:

```
►►──PORTCLNT──┬────┬──────────────────────────────────────────────────►◄
              └─-r─┘
```

Specify PORTCLNT command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

-r Specifies that the RPIVAL command is to be used to validate user IDs and passwords. It is recommended that you do not specify this option using the `:Parms.` tag, but instead specify `:ESM_Enable.YES` in the DTCPARMS file. For more information on using an external security manager, see "Appendix A. Using TCP/IP with an External Security Manager" on page 617.

Step 4: Provide NDB Agents Access to DB2 Server for VSE & VM

NDB uses the DBUTIL2 program to interface with DB2. The DB2-internal representation of DBUTIL2 is shipped with TCP/IP as DBUTIL2 ACC_OUT and contains a DB2 access module.

The access module must be loaded by the user ID that built the module. The module included with TCP/IP was built by TCPMAINT and must be loaded by TCPMAINT. The procedure to load or reload the access module is:

1. Log on to TCPMAINT and issue the following command:
   ```
   ndbinit tcpmaint
   ```
2. Contact the DB2 Database Administrator for your installation to issue the following DB2 command. This SQL command allows you to execute the DBUTIL2 program. The DB2 command syntax is:
   ```
   grant run on tcpmaint.dbutil2 to public
   ```

The NDB agent servers require access to the tables in the DB2 database to allow NDB clients to issue DB2 commands against these tables. The NDB agent user ID is passed to DB2 for processing a unit of work. Ask your DB2 Database Administrator or the owner of the tables to which the NDB clients need access, to issue the following DB2 command:

```
grant privilege on table to user_id
```

where *privilege* specifies any of the DB2 table privileges such as **select**, **update** and **all**; *table* specifies the table in the DB2 database that an NDB client accesses; and *user_id* specifies the user ID of one or more NDB agent servers. All NDB agents should be granted identical privileges because a one-to-one connection is created between an NDB client and an NDB agent for a given unit of work (UOW), and the NDB client cannot control which NDB agent is used for that UOW.

Before using the NDB agents the first time, initialize them for using DB2. This can be done by logging onto each NDB agent server and issuing the following commands:

```
set lang (add ari user
SQLINIT dbname(dbname)
```

where *dbname* is the desired DB2 database.

For more information about DB2 commands and granting privileges on tables in DB2, see the *DB2 Server for VSE & VM: SQL Reference*.

Chapter 17. Configuring the NFS Server

The NFS server implements the Network File System (NFS), as well as PCNFSD user ID authentication function. To configure the NFS server, you must perform the following steps:

NFS Server Configuration Steps

1. Enable the NFS server feature.
2. Update the TCPIP server configuration file.
3. Update the DTCPARMS file for the NFS server.
4. Establish NFS server machine authorizations.
5. Customize the VMNFS CONFIG file.
6. Configure NFS server file translation support. (Optional)
7. Verify NFS server operations.
8. Perform advanced NFS server configuration, if needed.

Dynamic Server Operation

The NFS server provides a VM Special Message (SMSG) interface that allows you to perform server administration tasks through a set of privileged commands. For more information, see "Dynamic Server Operation" on page 433.

Step 1: Enable the NFS Server Feature

NFS server is an optional feature of z/VM Version 3 Release 1.0 that must be enabled before you use it. For information on enabling the NFS server feature, see the *TCP/IP Feature for z/VM, Level 3A0 Program Directory*.

Step 2: Update PROFILE TCPIP

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

```
AUTOLOG
  VMNFS 0
```

The NFS server requires that ports TCP 2049 and UDP 2049 be reserved for it. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
  2049 UDP VMNFS           ;  NFS  Server
  2049 TCP VMNFS NOAUTOLOG  ;  NFS  Server
```

Step 3: Update the DTCPARMS File

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

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

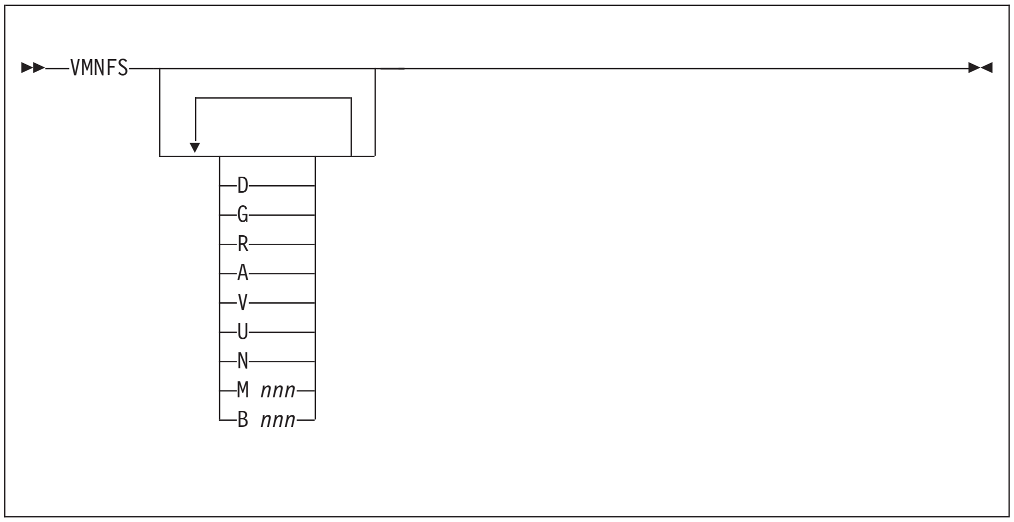
If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the NFS server if you:
- Enable access to the server without requiring a VM user ID and password (anonymous access).
- Use an External Security Manager (ESM) for client authentication and minidisk access control.
- Access an SFS directory where trace information will be written using the SMSG TWRITE command.
- Change the CMS SET RECALL setting for the NFS server virtual machine to allow access to migrated SFS and BFS files.

VMNFS Command

NFS services are initiated using the VMNFS command:



Specify VMNFS command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

D A debug flag that causes messages to be sent to the console. This operand also causes a log file to be generated as if the (**G**) operand were specified.

G Causes a binary output file named VMNFS LOG A1 to be generated that contains a record of all RPC requests received and replies sent. Any existing log file is erased when the NFS server is started.

R Indicates an external security manager (ESM) is to be used for access control. It is recommended that you specify `:ESM_Enable.YES` in the DTCPARMS file instead of providing this operand as a `:Parms.` tag startup parameter.

A Causes Autolink access control to be used for mount requests.

V Forces the NFS server to accept only Version 2 (RFC 1094) requests.

U Forces the NFS server to accept only UDP connections. TCP connections are not permitted.

N Indicates that ANONYMOUS mounts are permitted. It is recommended that you do not specify this operand as a `:Parms.` tag startup parameter, but instead specify `:Anonymous.YES` in the DTCPARMS file.

M *nnn*
Sets the trace mask to *nnn*, where where *nnn* is a decimal number. The default trace mask is zero. For information about using trace masks to diagnose NFS server problems, consult the *TCP/IP Diagnosis Guide*.

B *nnn*
Sets the number of disk blocks, where *nnn* is a decimal number. The default number of disk blocks is 256.

Using an External Security Manager

The NFS server can use an external security manager (ESM) to authenticate NFS clients and to control access to minidisks by specifying `:ESM_Enable.Yes` in the DTCPARMS file. For more information, see “Appendix A. Using TCP/IP with an External Security Manager” on page 617.

External Security Managers can protect SFS and BFS resources. However, that protection occurs in the file pool server machine, not in the NFS server. See *z/VM: CMS File Pool Planning, Administration, and Operation* for more information.

Step 4: Establish NFS Server Machine Authorizations

The system (CP) directory entry for the NFS server virtual machine must have **OPTION DIAG88** and **privilege class B** specified.

For NFS clients to access files or directories in the CMS Shared File System (SFS), the NFS server must have SFS file pool administrator authority. Each NFS server that will provide such access must be listed on the ADMIN statement in the SFS file pool server’s DMSPARMS file. For details on SFS file pool configuration and administrator authority, see *z/VM SFS and CRR Planning, Administration, and Operation*.

For NFS clients to access files and directories in the Byte File System (BFS), the NFS server must have connect authority to the file pool, and the NFS server must be defined as a POSIX "superuser". To allow this capability, the following statements must be included in the CP directory entry for the NFS server virtual machine:

```
POSIXINFO UID 0 GID 0
POSIXOPT QUERYDB ALLOW
```

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

Step 5: Customize the VMNFS CONFIG File

The NFS server configuration file, VMNFS CONFIG, contains configuration parameters for the NFS server. The statements used in this file specify:

- whether the PCNFSD function is available on the NFS server.
- whether NFS clients can request a list of all mounted file systems.
- the definition of the export list.
- whether NFS clients can mount other than what is defined in the export list.
- how many NFS clients using the TCP transport protocol can be concurrently handled by the NFS server.

See "NFS Configuration File Statements" for detailed information about how to specify entries within this file.

NFS Configuration File Statements

NFS configuration file statements are processed when the NFS server is started. If you change one of these statements while the NFS server is in operation, the change will not be effective until the NFS server is restarted or an SMSG M REFRESH CONFIG is issued to the NFS server.

If the VMNFS CONFIG file cannot be found or cannot be opened, NFS server initialization continues, using the following default values:

- when the `lines=ext` or `trans=ext` keywords are used on a MOUNT, the default file extension values used are as defined by the TCPIP DATA file. See the "Chapter 5. Defining the TCP/IP System Parameters" on page 139 for detailed information.
- the PCNFSD function is available on the VMNFS server
- NFS clients can obtain the list of all currently mounted file systems.
- There are no items in the export list.
- The maximum number of concurrent NFS clients using the TCP transport protocol is 50.

If the VMNFS CONFIG file cannot be read, the NFS server terminates with an error message.

Syntax Rules

The following syntax rules apply to statements specified in the NFS configuration file:

- Keywords (eg. EXPORT, etc.) are treated as if they were entered in uppercase.
- Variable operands are treated as if they were entered in uppercase *except* for EXPORT records, for which these operands are **case sensitive**.
- Configuration statements must begin and end on the same line.
- All comments must be preceded by a semicolon (;). A comment may follow a complete keyword and data specification on a record, or it may occupy a complete record.
- Blank records may be used to improve readability.

DUMPMOUNT Statement

The DUMPMOUNT statement determines whether the NFS server should make available to clients a list of all mounted file systems.

```
       ┌─DUMPMOUNT──YES─┐
►►─────┤                ├──────────────────────────────────────────────►◄
       └─DUMPMOUNT──NO──┘
```

Operands

YES
Indicates the list of all mounted file systems is to be made available to clients. This is the default if a DUMPMOUNT statement is not specified in the NFS configuration file.

NO
Indicates the list of all mounted file systems is *not* to be made available to clients.

Usage Notes

1. An NFS client requests the list of mounted file systems from the VMNFS server by sending a `MNTPROC_DUMP` request. If DUMPMOUNT YES is configured, the `MNTPROC_DUMP` reply contains the requested information.
2. DUMPMOUNT YES may make available to NFS clients the names of the SFS and BFS directories in your file pools. If you do not want to make this information available to any NFS client who can connect to your NFS server, specify DUMPMOUNT NO in the NFS server configuration file.
3. The information returned in a `MNTPROC_DUMP` reply includes resources actively in use by the VMNFS server. SFS and BFS directories are considered active if used within 15 minutes. A minidisk is considered active as long as the NFS server has that minidisk linked.

EXPORT Statement

The EXPORT statement defines an entry to be added to the NFS “export list”, which consists of all EXPORT statements defined in the NFS configuration file. An NFS client can obtain the export list by using the `OPENVM SHOWMOUNT` command.

```
►►──EXPORT──export_name──┬──────────────┬──────────────────────────────►◄
                         └─mount_string─┘
```

Each EXPORT statement consists of a symbolic name that will be presented to NFS clients, optionally followed by the string that will be used when the NFS server receives a mount with a symbolic name.

Operands

export_name
: The file system name that an NFS client can use to mount the *mount_string* (specified as the next operand). The *export_name* operand is **case sensitive** and is treated by the NFS client as a file system path name.

mount_string
: An optional parameter. The *mount_string* identifies the file system to be mounted and any mount options that are to be substituted when the NFS server receives a mount request for *export_name*. The *mount_string* must be a syntactically valid mount command; that is, the object to be mounted (minidisk, SFS or BFS directory) followed by any options for the mount.

  To assist in producing a valid *mount_string* the NFS server will recognize and process the following keywords within a *mount_string*:

%USERID	Causes the user ID that was specified on a call to the PCNFSD or on the `userid=` parameter of the MOUNTPW procedure to be substituted in place of the **%USERID** keyword.
%FSROOT	Causes the FSROOT parameter of the POSIXINFO CP directory statement to be substituted in place of the **%FSROOT** keyword. The user ID specified on a call to the PCNFSD or on the `userid=` parameter of the MOUNTPW procedure will be used to look up the VM CP directory entry.
%IWDIR	Causes the IWDIR parameter of the POSIXINFO CP directory statement to be substituted in place of the **%IWDIR** keyword. The user ID specified on a call to the PCNFSD or on the `userid=` parameter of the MOUNTPW procedure will be used to look up the VM CP directory entry.

Usage Notes

1. If the EXPORTONLY YES statement is specified, then only the export list defined by the EXPORT records in the NFS configuration file can be mounted by NFS clients.
2. If an EXPORT statement does not specify the *mount_string* parameter, then the *export_name* must be a syntactically valid mount command with options, if any.
3. Due to operating system-specific conventions for displaying path names to files, it is recommended that an *export_name* consist of path name components that are eight or less characters. For example, use `/PC/Your/SFS/In/FPCOOL` as opposed to `PC/Your_SFS_In_FPCOOL`.
4. For examples of EXPORT statements, consult the sample NFS configuration file, VMNFS SCONFIG.
5. To export BFS directories whose names include spaces, place the *mount_string* within double quotation marks("). An export name can also include spaces if it is surrounded by double quotation marks. Keep in mind that some NFS clients require special syntax to mount an alias with quotes and/or spaces. For example, one client requires the following syntax to mount the directory named “my directory” to mydir:

   ```
   mount gd3vm0:\"my\directory\" mydir
   ```
6. The NFS server ignores any export entries that have aliases which have already been used by another export entry. Initial slashes (/) are interpreted as being nonexistent, so the aliases “/alias” and “alias” are considered equivalent.

EXPORTONLY Statement

The EXPORTONLY statement restricts the file systems that can be mounted by NFS clients to the list of EXPORT records defined in the NFS configuration file.

```
      ┌─EXPORTONLY──NO──┐
►►────┤                 ├──────────────────────────────────────────►◄
      └─EXPORTONLY──YES─┘
```

Operands

NO
Indicates there are no restrictions on the file systems that can be mounted by an NFS client. In this case the export list augments what can be mounted. This is the default if the EXPORTONLY statement is not specified in the NFS configuration file.

YES
Indicates that NFS clients can mount only those file systems which are identified by EXPORT statements within the NFS configuration file.

Usage Notes

1. If EXPORTONLY YES is specified but no EXPORT records are defined, **and**:
   - the NFS server is being initialized with this configuration, the server terminates with an error message.
   - this configuration is put into place using an SMSG M REFRESH CONFIG command, then the NFS server will not allow any new mounts. However, the use of previously mounted file systems remains unaffected. A warning message is displayed on the server console indicating that the NFS server is in this state. To resume handling new mount requests, a new NFS configuration file should be reloaded in which either EXPORTONLY NO or an export list is defined.

MAXTCPUSERS Statement

The MAXTCPUSERS statement specifies the maximum number of NFS clients using the TCP transport protocol that can be concurrently supported by the NFS server.

```
      ┌─MAXTCPUSERS──50─────────────────┐
►►────┤                                 ├──────────────────────────►◄
      └─MAXTCPUSERS──maxtcpusers_value──┘
```

Operands

maxtcpusers_value
Defines the value to be used. If a MAXTCPUSERS statement is not specified in

the NFS configuration file, the NFS server defaults to supporting 50 concurrent NFS clients that use the TCP transport protocol.

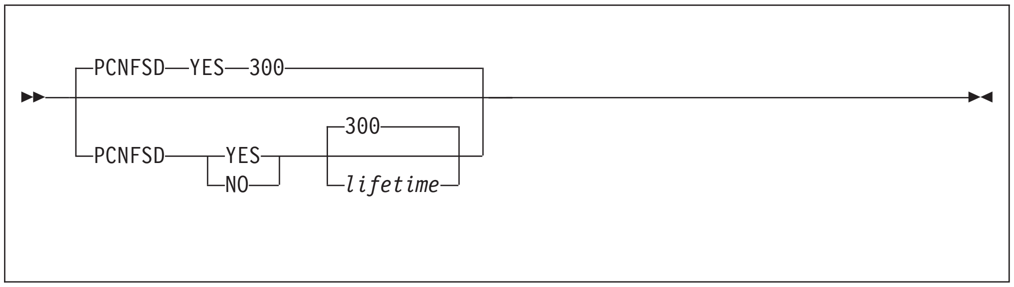
Usage Notes

1. The SMSG M REFRESH CONFIG command cannot be used to put a new *maxtcpusers_value* into effect while the NFS server is in operation. To activate the new value, the NFS server must be stopped and restarted.
2. For installations whose NFS clients are configured to use the TCP transport protocol, the default *maxtcpusers_value* may need to be increased. Otherwise, some of these users may be unable to access the NFS server. They may experience errors indicating connections were refused or not allowed. Since it may be difficult to determine how many users rely on use of the TCP transport protocol, it may be desirable to initially configure an arbitrarily large *maxtcpusers_value* and then revise this value over time.

   To assist in tuning the *maxtcpusers_value*, an SMSG M QUERY CONFIG command can be used to query the *maxtcpusers_value* currently in effect and the current number of NFS clients using TCP transport protocol. Note that values returned in the response reflect the current instance of the NFS server since it was started or restarted. For more information about this SMSG command, see the chapter, "Using the Network File System Commands", in the *TCP/IP User's Guide*.
3. The MAXTCPUSERS value used by the NFS server may be lower than the *maxtcpusers_value* specified. Message DTCNFS1553I is displayed on the VM NFS server console during its initialization to indicate this has occurred. There are two reasons why this can occur:
   - The NFS server requires a small number of socket connections for its use. Therefore, if *maxtcpusers_value* prevents the NFS server from obtaining these connections, MAXTCPUSERS will be set to a lower value.
   - The number of socket interface control blocks (SKCBs) specified with the SKCBPOOLSIZE statement in the TCPIP server configuration file may need to be increased in order for the desired *maxtcpusers_value* to be used.

PCNFSD Statement

The PCNFSD statement specifies whether PCNFSD support is to be made available by the NFS server.



Operands

YES
Indicates that PCNFSD support is made available by the NFS server. The default is PCNFSD YES 300, which causes PCNFSD information to be discarded after 5 minutes (300 seconds). Specifying PCNFSD YES removes the

need for a user ID and password to be supplied on mountpw and mount requests that are submitted by clients that have PCNFSD support.

NO
Indicates that PCNFSD support is to not be made available by the NFS server.

lifetime
The number of seconds the NFS server should keep PCNFSD information active after a PCNFSD request is received from an NFS client. The default is 300 seconds (5 minutes). After the specified *lifetime*, the PCNFSD information is discarded.

VMFILETYPE Statement

The VMFILETYPE statement is supported within the NFS server configuration file only to maintain compatibility with prior levels of TCP/IP for z/VM. File extension support should be configured through use of an equivalent VMFILETYPE statement within the TCPIP DATA file.

For detailed information about VMFILETYPE operands, see "Chapter 5. Defining the TCP/IP System Parameters" on page 139.

Note: The syntax for a VMFILETYPE statement within the TCPIP DATA file differs slightly compared to that for the NFS server configuration file.

Step 6: Configure NFS Server File Translation Support (Optional)

By default, the NFS server does not manipulate the file data it processed in response to client requests — that is, no EBCDIC-ASCII data translation is performed and line feed characters are not inserted at CMS record boundaries when files are processed. However, the NFS server can be configured to perform these actions for specific types of files, based on a file *extension* (or with respect to CMS files, the file *type*) of a file that is referenced. This can simplify NFS operations for various users and clients, and may even be necessary to accommodate certain NFS clients.

To configure file translation support for the NFS server, customize the TCPIP DATA file to include the appropriate VMFILETYPE and VMFILETYPEDEFAULT statements. The NFS server relies upon these statements to control the manner in which file translation and line feed processing are performed for specific file extensions, as well as those that are "unknown" or not recognized. For detailed information about how to specify these statements, see "Chapter 5. Defining the TCP/IP System Parameters" on page 139.

Notes:

1. The VMFILETYPE statement determines whether EBCDIC-ASCII translation occurs for a specific file and whether line feed characters are inserted at CMS record boundaries, based on the extension of that file.
2. File extensions that are not dealt with through a specific VMFILETYPE statement are considered as "unknown" (that is, are not recognized). The translation performed for such files (if any) is controlled by the VMFILETYPEDEFAULT statement. If the VMFILETYPEDEFAULT statement is not used, no translation is performed and no line feed characters are inserted at CMS record boundaries.

3. Case (upper or lower) is not significant when the NFS server compares the *filetype* (*extension*) supplied in an NFS request with those on *filetype* VMFILETYPE statements. For example, `BIN` is considered to be equivalent to `Bin`.

For additional information about how NFS clients can make use of file translation, see the *TCP/IP User's Guide*.

Step 7: Verify NFS Server Operations

After the NFS server has successfully initialized, use the steps that follow to verify that the server is configured and running correctly:

1. From another system, issue a PING command against the z/VM host to verify that network connectivity has been established. For example:
```
ping vmsys1.endicott.ibm.com
```
2. Issue an RPCINFO command to verify that the Portmapper server is up and running. The RPCINFO command can be issued from any platform that supports this command. For a z/VM host, the RPCINFO command format is as follows:
```
rpcinfo -p server_name
```
   For example:
```
rpcinfo -p vmsys1.endicott.ibm.com
```
   The Portmapper service should respond with a list of programs, versions, protocols, and port numbers. This response should be similar to the following:
```
rpcinfo -p vmsys1
   program vers proto   port
    100000    2   udp    111  portmapper
    100000    2   tcp    111  portmapper
    100005    1   udp   2049  mountd
    100005    3   udp   2049  mountd
    100005    1   tcp   2049  mountd
    100005    3   tcp   2049  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    150001    1   udp   2049  pcnfsd
    150001    2   udp   2049  pcnfsd
Ready; T=0.04/0.08 16:38:21
```
   If a similar response is not returned, start the Portmapper server or have this done by the appropriate system administration personnel.
3. If your users mount using the NFS Version 3 or TCP transport protocol, verify that the output from the RPCINFO -p command lists **3** in the `vers` column and **tcp** in the `proto` column for both the **mountd** and **nfs** programs.
4. Verify that the **mountd**, **pcnfsd**, **portmapper**, and **nfs** services are operating correctly on your VM system by entering the following VM RPCINFO commands:
```
rpcinfo -u host_name mount
rpcinfo -u host_name pcnfsd
rpcinfo -u host_name portmapper
rpcinfo -u host_name nfs
```
   If these services are running on the VM system, the following responses are returned:

```
Ready; T=0.02/0.06 16:45:36
* See if mount is running
rpcinfo -u vmsys1 mount
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
Ready; T=0.04/0.08 16:45:49

* See if portmapper is running
rpcinfo -u vmsys1 portmapper
program 100000 version 2 ready and waiting
Ready; T=0.04/0.08 16:46:40

* See if nfs is running
rpcinfo -u vmsys1 nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
Ready; T=0.04/0.08 16:47:27

* See if pcnfsd is running
rpcinfo -u vmsys1 pcnfsd
program 150001 version 1 ready and waiting
program 150001 version 2 ready and waiting
Ready; T=0.04/0.08 16:47:53
```

5. Test the NFS server by issuing a mount request from a remote client. If the NFS server terminates with a return code of 101 or 102, a request to allocate a control block (101), or a disk block (102) buffer failed.
6. Verify the correct VMNFS CONFIG file is in effect and that this file has been customized as intended. To do this, issue the following command:

   ```
   server_id m query config
   ```

   to the NFS server from an appropriately authorized user ID, such as TCPMAINT. The response for this command can be reviewed to verify that the EXPORTONLY statement has been correctly specified to allow or disallow users from performing mounts. A sample SMSG QUERY CONFIG command response follows:

```
smsg vmnfs m query config
Ready; T=0.01/0.01 16:39:54
 16:39:54  * MSG FROM VMNFS  : M Exportonly no, Anonymous yes, Dumpmount yes, Security CP
 16:39:54  * MSG FROM VMNFS  : M PCNFSD yes, time out after  300 seconds
 16:39:54  * MSG FROM VMNFS  : M UDP V3 buffer sizes: 8192 read, 8192 write, 8192 preferred.
 16:39:54  * MSG FROM VMNFS  : M TCP V3 buffer sizes: 65536 read, 65536 write, 32768 preferred.
 16:39:54  * MSG FROM VMNFS  : M Maximum TCP clients: 50.
 16:39:54  * MSG FROM VMNFS  : M No NFS clients are currently using TCP.
```

Step 8: Advanced Configuration Considerations

Before you complete the NFS server configuration process, you may want to review the information in the following sections to determine if additional NFS configuration is appropriate or necessary for your installation.

NFS Server Exits

Several server exits are supported for use with the NFS server that allow for greater control over client mount requests, as well as client and administrative interaction with the server. These exits are described in more detail in the sections that follow.

CMS Command Exit

The CMS command exit (VMNFSCMS EXEC) provides authorization and execution control over CMS commands issued to the NFS server through the SMSG interface.

For example, CMS commands similar to the following might be issued by the TCPMAINT user ID to obtain the NFS server console spool file while the server is operating:

```
cp smsg vmnfs m1 cms cp sp con to tcpmaint
cp smsg vmnfs m2 cms cp clse con
cp smsg vmnfs m3 cms identify
```

Keep in mind the following when the CMS command exit is customized:

Notes:

1. Any security guidelines or authorization controls established for your site should be incorporated within the exit before it is activated.
2. Consideration should also be given to traditional security issues when this exit is modified and enabled. For example, the NFS server requires the use of privileged commands that are denied to general users, and may have access to files that contain passwords (such as the VMNFS HISTORY file). Measures should be taken to ensure areas such as these are properly addressed.
3. Care should be taken so that commands or programs that may adversely affect the NFS server are not permitted. For example, a command that overlays the VMNFS MODULE or that causes CMS storage management changes should be restricted from use. Likewise, a program that uses TCP/IP services — even if it does not create a storage conflict — is likely to cause operational problems.
4. The NFS server utilizes the first command exit found in its CMS search order. If a CMS command exit is not available to the NFS server, it rejects all SMSG CMS commands.

CMS Command Exit Input: For each SMSG CMS command that it receives, the NFS server stores values pertinent to that command in several GLOBALV variables (maintained in the GLOBALV group **VMNFS**) which can be referenced within the CMS command exit. These variables, and a description of the value each may have, are listed here:

Variable	Description
ORIGIN_NODE	The name of the original sending host, if the SMSG is delivered through RSCS. An asterisk (*) is supplied if the command is not from RSCS.
ORIGIN_USERID	The virtual machine that originated the SMSG command, if the SMSG is delivered through RSCS. An asterisk (*) is supplied if the command is not from RSCS.
REPLY_TAG	The *replytag* supplied to the NFS server for this SMSG command.
VERB	The **CMS** operand of the SMSG command. For this exit, ’CMS’ is a constant that does not change.
ARGS	The CMS command arguments supplied with the SMSG CMS command.
SMSG	The complete, unmodified SMSG command text.

The following are passed to the CMS command exit as command line arguments:

Argument	Value	Format
1	CMS command arguments as supplied with the SMSG CMS command.	Character

Return Codes: The NFS server does make use of the return code established by this exit. However, a nonzero return code will cause an informational message and a list of GLOBALV variables and values to be displayed at the NFS server console.

CMS Command Exit Sample: A sample CMS command exit is provided as VMNFSCMS SEXEC on the TCPMAINT 198 minidisk. Your customized exit should be maintained on this same minidisk, with a file type of EXEC. For more information about the supplied CMS command exit, review the content of the VMNFSCMS SEXEC file.

Mount Monitor Exit

The mount monitor exit (VMNFSMON EXEC) provides the ability to monitor or control all client mount requests. The monitor exit allows these requests to be accepted or rejected based on information about the requesting user or client, such as the client host IP address. In addition, the monitor exit may modify certain mount string values supplied by a client before mount processing is completed.

Keep in mind the following when the mount monitor exit is customized:

Notes:

1. The NFS server utilizes the first mount monitor exit found in its CMS search order. If a mount monitor exit is not available to the NFS server, it processes all mount requests based only on client-supplied mount parameters.

Mount Monitor Exit Input: For each mount request that it receives, the NFS server stores values pertinent to that request in several GLOBALV variables (maintained in the GLOBALV group **VMNFS**) which can be referenced within the mount monitor exit. These variables, and a description of the value each may have, are listed here:

Variable	Description
ACCOUNT	The account identification string provided with the mount request via the `account=` operand.
CLIENT	The IP address (in dotted-decimal format) of the NFS client host system that issued the mount request. For example: 9.130.48.134
FILESYSTEM	The minidisk, SFS, or BFS resource to be mounted through this mount request. For example, `ELWOOD.191` might be supplied for a minidisk mount request, whereas `FPCOOL:ELWOOD.` might be supplied for an SFS mount, and `/../VMBFS:FPCOOL:ROOT/home/elwood/.` might be supplied for a BFS request.
LINKPASSWORD	The minidisk link password supplied with a minidisk mount request, via the `mdiskpw=` operand. For SFS and BFS mount requests, the LINKPASSWORD value is not applicable and is not defined.

RECORD
A decimal value that corresponds to the presence of `trans=`, `lines=`, and `record=` operands in the mount request. Possible RECORD variable values follow:

RECORD Value	Associated Mount Operands
136	`trans=ext` and `lines=ext`
132	`trans=ext` and `lines=nl`
130	`trans=ext` and `lines=CMS`
129	`trans=ext` and `lines=none`
72	`trans=yes` and `lines=ext`
68	`trans=yes` and `lines=nl`
66	either (`trans=yes` and `lines=CMS`) or (`record=text`)
65	`trans=yes` and `lines=none`
40	`trans=no` and `lines=ext`
36	`trans=no` and `lines=nl`
34	either (`trans=no` and `lines=CMS`) or (`record=binary`)
33	`trans=no` and `lines=none`

TYPE
A decimal value that corresponds to the type of mount request. Possible TYPE variable values follow:

TYPE Value	Mount Type
17	Read-write mount request
18	Read-only mount request

USERID
The logon user ID specified with the mount request via the `userid=` operand, or through a PCNFSD request.

BYUSERID
The LOGONBY user ID specified with the mount request via the `by=` operand.

Values for the ACCOUNT, FILESYSTEM and LINKPASSWORD variables can be modified within the mount monitor exit. Changes to values maintained by these GLOBALV variables are then used by the NFS server when the monitor exit returns processing control.

Note: If the FILESYSTEM value is altered, the *type* of file system that was specified by the requesting client must be maintained. For example, if FILESYSTEM originally represents a minidisk, any changed value must continue to represent a minidisk — an SFS or BFS directory cannot be substituted in such a case.

No command line arguments are passed to the mount monitor exit.

Return Codes: The NFS server recognizes the following exit return codes:

Return Code	Meaning
0	Continue and process the mount request
nonzero	Deny the mount request and return ″not authorized″ status to the client

Mount Monitor Exit Sample: A sample mount monitor exit is provided as VMNFSMON SEXEC on the TCPMAINT 198 minidisk. Your customized exit should be maintained on this same minidisk, with a file type of EXEC. For more information about the supplied mount monitor exit, review the content of the VMNFSMON SEXEC file.

SMSG Authorization Exit

The SMSG authorization exit (VMNFSSMG EXEC) provides the ability to control the acceptance and processing of SMSG commands issued to the NFS server, based on the type and purpose of an SMSG command, as well as the originator of the command.

Keep in mind the following when the SMSG authorization exit is customized:

Notes:

1. Any security guidelines or authorization controls established for your site should be incorporated within the exit before it is activated.
2. The NFS server utilizes the first SMSG authorization exit found in its CMS search order. If an SMSG authorization exit is not available to the NFS server, it accepts and processes all SMSG commands (with SMSG CMS command processing controlled by the CMS command exit, or lack thereof).

SMSG Authorization Exit Input: For each SMSG command that it receives, the NFS server invokes the SMSG authorization exit, with the following passed as command line arguments:

Argument	Value	Format
1	User ID that originated the SMSG command	Character
2	Node ID from which the SMSG command was issued	Character
3	Text of the supplied SMSG	Character

Return Codes: The NFS server recognizes the following exit return codes:

Return Code	Meaning
0	Accept and process the supplied SMSG command.
nonzero	Do not process the supplied command — the originating user ID is not authorized.

SMSG Authorization Exit Sample: A sample SMSG authorization exit is provided as VMNFSSMG SEXEC on the TCPMAINT 198 minidisk. Your customized exit should be maintained on this same minidisk, with a file type of EXEC. For more information about the supplied SMSG authorization exit, review the content of the VMNFSSMG SEXEC file.

Managing Translation Tables

Many different translation tables can be made available to clients, in order to facilitate the proper ASCII/EBCDIC translation of data. The specific translation table to be used in processing data for a given client is specified as part of its MOUNT request, through use of the **xlate=***tablename* operand. If the corresponding *tablename* TCPXLBIN file is present in the search order for the NFS server, that translation table is then used; if such a file is not available, the mount attempt fails. See "Chapter 29. Using Translation Tables" on page 595 for more information about using translation tables.

A maximum of 255 translation tables can be accommodated by the NFS server. If this maximum is exceeded, a notification message is displayed at the NFS server console and an I/O error is returned to any client that requests the use of a translation table which is not already active.

The mapping of translation tables currently in use by the NFS server can be ″refreshed″ through the following actions:

1. Stop the NFS server
2. Erase the **VMNFS TRANSLAT** file that resides on the server 191 minidisk.
3. Restart the NFS server.

Notes:

1. Do not attempt to directly modify the **VMNFS TRANSLAT** file, as this can corrupt its content.
2. When the translation table mapping is refreshed in this manner, the **VMNFS HISTORY** file is also refreshed. This secondary action invalidates all current file handles and causes a ″stale handle″ notification to be returned to any client that attempts to use a previously-mounted file system. Thus, clients must remount any file systems that have been in use when this type of refresh operation is performed.

Allowing Access to Migrated SFS and BFS Files

By default, the CMS SET RECALL setting for the NFS server virtual machine is **OFF**, which indicates that a reference to data within an SFS or BFS file that is in *migrated status* (that is, has been moved to DFSMS/VM storage) will not cause that data to be implicitly recalled. With respect to the NFS server, the CMS SET RECALL OFF setting causes an `NFSERR_IO` error to be returned to a client when such a reference is made.

If you wish to change this setting to prevent clients from encountering this error, you must create a server profile exit for the NFS server. Within the NFS server exit, include the SET RECALL ON command in the **BEGIN** processing section of the exit. In addition, the DTCPARMS file must include an **:exit.** tag that identifies the NFS server profile exit.

Note: When the SET RECALL setting is changed to **ON**, NFS clients may encounter long delays when requests and references are made for data that is maintained in migrated files.

Managing Data Transfer Operations

Depending on the NFS protocol that is used by a client, it is possible to perform some customization that can affect how data transfer operations are completed by the NFS server.

If NFS version 2 protocol is used by a client, the NFS server uses a maximum of 8192 bytes for READ and WRITE data transfer operations when requests are satisfied. This is a limitation imposed by the NFS version 2 protocol specification that applies to both the UDP and TCP transport protocols.

For the NFS version 3 protocol, the NFS server uses data buffer values that are defined for the TCP/IP stack to determine its maximum and preferred READ and WRITE data transfer sizes. The UDP values used by the NFS server are based on the LARGEENVELOPEPOOLSIZE *lrg_env_size* value that is specified in the TCP/IP sever configuration file. The TCP values used are based on the DATABUFFERPOOLSIZE *buffer_size* value (up to a maximum of 65,536 bytes) that is defined within this same file. For both UDP and TCP, the actual data transfer sizes used by the NFS server may be slightly less than their corresponding TCP/IP server configuration values, due to the inclusion of RPC header information that is required on all data transfers.

When NFS version 3 protocol is in use, NFS clients can determine the maximum and preferred READ and WRITE data transfer sizes that have been established for the NFS server (and which govern its capabilities). This information can then be used by clients to help determine the data transfer sizes they should specify when requests are made of the NFS server.

Adjusting TCP/IP Data Buffers

To determine the NFS version 3 maximum and preferred data sizes that are currently in use by the NFS server, issue the following SMSG command from an appropriately authorized user ID, such as TCPMAINT:

```
smsg server_id m query config
```

A sample response for this command follows, in which the UDP maximum and preferred READ and WRITE data transfer sizes are each 8784 bytes, while for TCP they are 7784 bytes:

```
NFS server - VM TCP/IP Level 3A0
Exportonly no, Anonymous yes, Dumpmount yes, Security ESM
PCNFSD yes, time out after  300 seconds
UDP V3 buffer sizes: 8784 read, 8784 write, 8784 preferred.
TCP V3 buffer sizes: 7784 read, 7784 write, 7784 preferred.
Maximum TCP clients: 50.
No NFS clients are currently using TCP.
```

If the data transfer sizes in use are not satisfactory, the following steps can be used to change their values:

1. Stop the TCP/IP server.
2. Modify the appropriate values for the LARGEENVELOPEPOOLSIZE and DATABUFFERPOOLSIZE statements in the TCP/IP server configuration file.
3. Restart the TCP/IP server.
4. Restart the NFS server.
5. Issue the SMSG previously cited SMSG command to verify the new data transfer sizes are in effect.

Keep in mind the following when you consider whether to change data transfer size values:

1. Changes to LARGEENVELOPEPOOLSIZE and DATABUFFERPOOLSIZE statement values affect **all** TCP/IP users, not just the NFS server.
2. Current users of TCP/IP services will be affected when the TCP/IP and NFS servers are stopped and restarted.
3. Additional virtual storage may need to be defined for the TCP/IP server virtual machine when larger envelope or data buffer sizes are specified.

Managing File Handle Operations

File Handle Overview

A *file handle* is a data structure used by NFS to identify a particular file that is to be used. File handles are generated by the NFS server in response to client requests, such as MOUNT and LOOKUP FILE. A client saves this file handle, and returns it when it issues subsequent requests (for example, READ FILE) that pertain to a given file. The data present in a file handle is meaningless to the client system.

For a VM/ESA minidisk, file-level control mechanisms do not exist — only disk-level access controls are present. If a minidisk can be mounted, *all* files on that minidisk are accessible to a client. Put another way, the file handle returned for a minidisk MOUNT represents the capability to access *any* file on the mounted disk. By comparison, the VM/ESA Shared File System (SFS) and Byte File System (BFS) include inherent file and directory-level control mechanisms. Thus, the file handles returned to clients when these file systems are referenced are provide file-level control.

The VMNFS HISTORY File

Because the 32-bytes that comprise a file handle are not adequate for identifying VM/ESA files, the NFS server maintains information about the file handles it distributes to clients within a file on its 191 minidisk — the VMNFS HISTORY file. Each file handle generated by the server is associated with record within this file. If a file handle received by the server designates a file that is not immediately known, the appropriate record from this history file is retrieved so that information needed to satisfy a request (such as minidisk link and control block structure information) can be obtained.

Note: Be aware that the VMNFS HISTORY file contains passwords. Therefore, access restrictions appropriate for your installation should be placed on the NFS server 191 minidisk.

Managing the VMNFS HISTORY File

Based on how NFS services are used within your environment (and due to system management operations that may affect the NFS server), actions may at times be necessary to manage certain aspects of the VMNFS HISTORY file.

Accommodating a Large User Population: Within the VMNFS HISTORY file, one record is used for each distinct VM user ID that accesses a minidisk, and one record is used for each distinct VM user ID that accesses one or more SFS or BFS directories in a given file pool. Because of this, the VMNFS HISTORY file may need to be enlarged to accommodate installations that have a large number of users. If this is necessary, the VMNFS HISTORY file must be increased to include additional records, all of which must contain only binary zeros. This can be done by appending the VMNFS HSITORY file that resides on the TCPMAINT 591 minidisk to the active history file present on the NFS server 191 minidisk. For example, assuming that one is logged on to the NFS server user ID and the TCPMAINT 591 minidisk is accessed at file mode E, the command:

```
copyfile vmnfs history e = = a (append
```

will add an additional 640 records to the active file on the NFS server A-disk.

Accommodating System Management and Similar Changes: On rare occasions a file pool administrator may find it necessary to recreate (**FILESERV GENERATE**) and reload all of the data maintained by a file pool server. However, this action can create problems if NFS clients attempt to use file handles that were created prior to the generation of a file pool, since file handles contain encoded pointers to the objects within a file pool. For example, after a FILESERV GENERATE operation, it is likely that a previously existing file pointer now points to a different file, or even to a directory. For this reason, it is necessary to ensure that all NFS-mounted SFS directories are unmounted before any file pools that contain those directories are (re)generated.

One method of forcing all mounted SFS directories to be unmounted is to stop the NFS server and erase the existing VMNFS HISTORY file. However, this approach also forces any mounted BFS directories or CMS minidisks to be unmounted.

Note: It is recommended that the VMNFS HISTORY file be erased whenever a new TCP/IP Function Level is installed.

Managing Old and Invalid File Handles: Once a file handle has been generated, it has no intrinsic life span, so it is valid indefinitely. However, a simple way of ensuring that file handles have a limited life span is to stop the NFS server and erase the active VMNFS HISTORY file at regular intervals. This forces clients to remount referenced file systems in order to receive new, valid file handles.

Notes:
1. When the VMNFS HISTORY file is erased, the **VMNFS TRANSLAT** file is also refreshed when the NFS server is again initialized.
2. There is no security exposure due to attempts to use old file handles; unauthorized access to data continues to be restricted such cases.

Using Additional Security Capabilities

File Handle Encryption (NFSFHCIP ASSEMBLE File)

To guard against the possibility that a client has modified or devised a file handle to gain unauthorized access to a file, the NFS server has the ability to encrypt all file handles. When enabled, each file handle that is generated by the NFS server in response to a client request is encrypted before it is provided to the client. When a file handle is returned to the NFS server, it is decoded to obtain the original structure that identifies the associated file. If the server detects a tampered encryption, the decoding process generates an invalid file handle and rejects the accompanying request.

Note: File data is not encrypted by the NFS protocol, and the NFS server does not have such capabilities. If data encryption is necessary for your environment, this must be performed independent of NFS server operations.

The NFSFHCIP ASSEMBLE file (Network File System File Handle Cryptographic Interface Program) can be used to invoke a cipher routine in order to encode and decode a file handle. As supplied, this file will default to invoking a cipher program named IPSASM, for which a corresponding IPSASM TEXT file must exist. A different cryptographic routine can be used in place of the IPSASM routine, though this may require minor changes to NFSFHCIP; for more information, see "Source Code Modifications".

Minidisk Link Monitoring (NFSBADPW C File)

When a client *minidisk* mount request is processed, an internal NFS server routine (NFSBADPW) is called after an attempt has been made to link the requested minidisk; this routine is called regardless of whether this link attempt succeeds or fails. As supplied with TCP/IP, this routine simply issues a message which contains details about the link failure to the NFS server console.

If additional action is desired or necessary when a link failure occurs (such as logging data in a disk file or informing the system operator or another user of a failure), the NFSBADPW routine can be updated or replaced, through modification of the NFSBADPW C program source file, which is provided as part of the TCP/IP Source Feature for z/VM.

Source Code Modifications

To make use of the security capabilities described in the previous sections, source code modifications are necessary which then require the NFS server program (VMNFS MODULE) to be rebuilt.

If file handle encryption is to be used by the NFS server, the following modifications are required:

1. Modify the NFSHCIP ASSEMBLE to employ calling conventions which are suitable for the encryption program that is to be used. The NFSFCHIP ASSEMBLE file can be found on the 3TCPIPA0 2B2 (Base code) minidisk, in the event the TCP/IP Source Feature for z/VM has not been installed.
2. If necessary, modify the TCPBLC91 EXEC (the VMSES/E build list that is used to build the VMNFS MODULE). Changes to this build list will required if the cipher routine text file (supplied with the encryption program in use) is not named, or cannot be renamed to, IPSASM TEXT. In such a case, the IPSASM TEXT entry within the TCPBLC91 EXEC must be changed to reflect the proper name.

If the NFSBADPW C file is to be updated or replaced, this file must be suitably modified and compiled, and the VMNFS MODULE must then be rebuilt.

For more information about making these modifications and rebuilding the VMNFS MODULE within the VMSES/E environment, see the *TCP/IP Feature for z/VM, Level 3A0 Program Directory*, beginning with the appendix that discusses VMNFS code modifications.

Dynamic Server Operation

The VM Special Message Facility (SMSG) command provides an interactive interface to the NFS server to:

- modify NFS server configuration attributes
- obtain various types of information about server operations and client mounts
- instruct the server to perform certain actions.

Note: The NFS server SMSG authorization exit can be used to control the acceptance and processing of SMSG commands issued to the NFS server. For more information, see "SMSG Authorization Exit" on page 427.

For detailed information about user-oriented SMSG commands that are supported by the NFS server, see the section titled *SMSG Interface to VMNFS* in the *TCP/IP User's Guide*.

SMSG Interface to the NFS Server

```
►►──SMSG──server_id──replytag──┤ Options ├──────────────────────────────────►◄

Options:

├───┬─Refresh────CONFIG───────┬──────────────────────────────────────────────┤
    │      ┌──────────────┐   │
    │      ▼              │   │
    ├─CMS──────arg────────┴───┤
    └─TWRITE──────────────────┘
```

SMSG command operands are parsed by the NFS server as blank-delimited tokens, and case is not significant. Any tokens present after those recognized by the NFS server are considered to be comments and are ignored.

Operands

server_id
The user ID of the NFS server virtual machine, usually VMNFS.

replytag
A character string that associates the supplied SMSG command *Option* with a server response; the *replytag* prefaces each message issued by the NFS server in response to the given command. Any characters (other than blank and null characters) can be used to form a *relpytag*, and case is not significant.

The *replytag* also indicates how the NFS server is to deliver its response to a particular SMSG command. The last character of *replytag* has special meaning and is interpreted by the server as follows:

The following describes how the last character of *replytag* is interpreted.

s	respond using the CP SMSG command.

m respond using the CP MSG command.

n send no response.

If the *replytag* does not end with one of these characters, the NFS server chooses a response mode.

Refresh CONFIG
Directs the NFS server to replace existing configuration information with that defined by current definitions in the NFS server configuration file (VMNFS CONFIG).

Notes:

1. Except for the MAXTCPUSERS statement, all records within the NFS configuration file are used to update NFS server configuration parameters. If a supported configuration statement is not present within this file, the default for that statement is used. For example, if the PCNFSD statement is omitted, the default value of YES is used.
2. The NFS server configuration file typically resides on the TCPMAINT 198 minidisk. For changes to this file to become effective, the NFS server must reaccess this minidisk prior to receipt of an SMSG REFRESH CONFIG command. A reaccess of the TCPMAINT 198 minidisk can be accomplished by issuing an appropriate SMSG CMS command to the NFS server, such as:
   ```
   smsg server_id m access 198 d
   ```
3. If changes are made to the NFS configuration file and the SMSG REFRESH CONFIG command is not used, those changes become effective when the NFS server is again initialized.

CMS *command_arguments*
Directs the NFS server to pass the supplied *command_arguments* to CMS for execution. Note that all commands are processed under control of the VMNFSCMS CMS command exit.

TWRITE
Causes trace tables to be written to file TRACEV FILE on the server A-disk.

Chapter 18. Configuring the Portmapper Server

The Portmapper (PORTMAP) server application is used to map program numbers and various numbers to RPC programs that request information. To configure the Portmapper server virtual machine, you must perform the following steps:

Portmapper Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the Portmapper server.
3. Verify Portmapper services.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
 PORTMAP   0
```

Note: If your system is using the Network File System (NFS) server, the Portmapper server must be started before the NFS server. If the NFS server does not receive a response from a Portmapper request, it waits a period of time for the Portmapper server to complete initialization,then tries again.

The Portmapper server requires port UDP 111 to be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
PORT
  111 UDP PORTMAP             ; Portmapper Server
```

Step 2: Update the DTCPARMS File

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

```
:Nick.PORTMAP
  :Parms.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the Portmapper server if you change the user ID of the virtual machine that receives the Portmapper console output.

PORTMAP Command

Portmapper services are initiated using the PORTMAP command:

```
►►──PORTMAP──┬────┬──────────────────────────────────────────────────────►◄
             └─-d─┘
```

Specify PORTMAP command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

-d Turns on debug tracing. Trace information is directed to the Portmapper server console.

Step 3: Verify Portmapper Services

To verify that Portmapper services are available:

1. If necessary, link and access the TCPMAINT 592 client code disk.
2. Issue the following command:

   ```
   rpcinfo -p loopback
   ```

   At a minimum, the response to this command should include the following:

   ```
   program vers proto   port
    100000    2   upd    111  portmapper
    100000    2   tcp    111  portmapper
   ```

Chapter 19. Configuring the REXEC Server

The REXEC server implements the Remote Execution Command protocol (REXEC). To configure the REXEC server virtual machine, you must perform the following steps:

REXEC Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the REXEC server.
3. Define additional REXEC agent virtual machines. (Optional)

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  REXECD   0
```

The REXEC server requires that TCP port 512 be reserved for it. Verify that the following statements have been added to your TCPIP server configuration file as well:

```
PORT
  512 TCP REXECD   ; REXEC Server
  514 TCP REXECD   ; REXEC Server
```

To allow the REXEC server to manage its agent virtual machines, it must be included in the OBEY list in PROFILE TCPIP. Verify that the following statements are added to PROFILE TCPIP:

```
OBEY
  REXECD
ENDOBEY
```

For more information about the OBEY list, see “OBEY Statement” on page 115.

Step 2: Update the DTCPARMS File

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

```
:Nick.REXECD
  :Parms.
  :Anonymous.
  :ESM_Enable.
  :ESM_Validate.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see “Chapter 3. General TCP/IP Server Configuration” on page 17.

Note: You should modify the DTCPARMS file for the REXEC server if you:
- Run the server with an External Security Manager (ESM), such as RACF.
- Choose to enable anonymous rexec capabilities.
- Override default command parameters for this server.

Step 3: Define Additional Anonymous REXEC Agent Virtual Machines (Optional)

To provide support for *anonymous* REXEC client requests, at least one REXEC *agent* virtual machine must be defined for your installation. The default TCP/IP installation environment includes one such agent virtual machine, named RXAGENT1.

Because anonymous REXEC requests are processed using only these agent virtual machines, the REXEC server can respond to such requests more readily when more than one agent machine is available.

For any additional REXEC agent machines that you define, it is recommended that you:
- maintain the RXAGENT*n* naming convention
- model your CP directory entries after that supplied for the RXAGENT1 virtual machine.

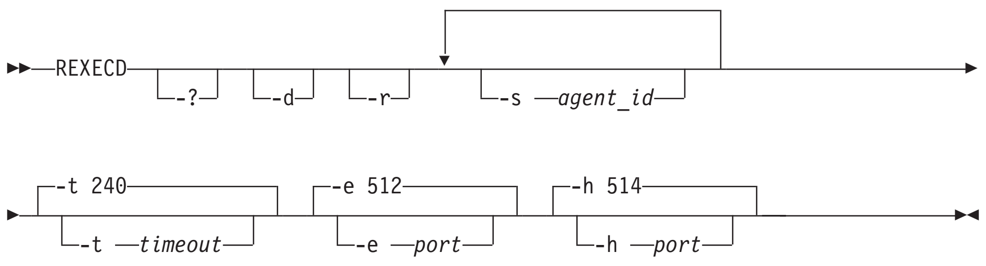
See “Chapter 3. General TCP/IP Server Configuration” on page 17 for more information about duplicating existing servers. If necessary, consult the *TCP/IP Function Level 320 Program Directory* for specific DASD storage and user ID requirements that may be applicable to these virtual machines.

Using an External Security Manager

The REXEC server can be configured such that client authentication will be under the control of an external security manager (ESM), such as RACF. For more information on using an ESM, see “Appendix A. Using TCP/IP with an External Security Manager” on page 617.

REXECD Command

REXEC services are initiated using the REXECD command:



Specify REXECD command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

-? Displays all options supported by the REXECD command.

-d Turns on debug tracing.

-r Indicates an external security manager is used to validate VM user IDs and passwords supplied by REXEC clients. It is recommended that you do not specify this option using the `:Parms.` tag, but instead specify `:ESM_Enable.YES` in the DTCPARMS file. For more information on using an external security manager, see "Appendix A. Using TCP/IP with an External Security Manager" on page 617.

-s *agent_id*
Identifies the *agent_id* virtual machine as a member of the anonymous client agent server pool. Repeat this parameter to identify any additional agent machines that will be used. It is recommended that you do not specify this parameter using the `:Parms.` tag, but instead specify `:Anonymous.YES` in the DTCPARMS file.

All agents must have `:Class.rexec_agent` and `:For.rexecd` (the user ID of the REXEC server) specified in the DTCPARMS file. A default agent virtual machine, RXAGENT1, is defined in this file.

See "Anonymous REXEC Client Processing" on page 440 for more information on how agents are used.

-t *timeout*
Sets idle time-out. This option specifies the time (in seconds) in which a connection closes if there is no activity. The default time-out is 240 seconds.

-e *port*
Sets the REXEC port. The default REXEC port is 512.

-h *port*
Sets the RSH port. The default RSH port is 514.

Additional REXEC Considerations

The sections that follow provide information about operational and processing characteristics inherent to the z/VM REXEC server implementation, which may help with the administration and use of REXEC services within a given environment.

How the REXEC Server Uses Secondary Virtual Machines

The REXEC server uses secondary virtual machines (agents) to execute commands passed from each ANONYMOUS or GUEST client. This allows the REXEC server to better service multiple REXEC requests, as the multiple agents accessible to the REXEC server help simulate a multitasking environment. Additionally, a user's own virtual machine can be used to process REXEC commands; when this is done, the user machine is managed somewhat differently than an agent machine.

Anonymous REXEC Client Processing

If `:Anonymous.YES` is specified in the DTCPARMS file, the REXEC server will maintain a "pool" of *agent* virtual machines to handle REXEC clients that log in as "anonymous" or "guest". When the REXEC server is started, there is a short delay (approximately 30 seconds) during which all agents in the pool are made ready; during this time, REXEC clients that attempt to use anonymous services will receive an indication that no agents are available. Once the agents are ready (logged on), the REXEC server can accept anonymous REXEC requests.

When an anonymous REXEC command is received, the command is sent to an available agent. That agent issues the command and returns the command response to the REXEC server, which then sends the response back to the REXEC client. When the transaction is complete, the agent is returned to the pool of available agents.

Agents are reinitialized by the REXEC server when:
- the REXEC server is itself reinitialized.
- an agent is determined to be inoperable, due to a problem in processing a command (that is, it does not return command output within the defined timeout period).

User's Own Virtual Machines

When a user's own virtual machine is used to execute an REXEC-supplied command, that machine is autologged by the REXEC server (by the XAUTOLOG command) after the supplied user and password have been authenticated. The given command is then processed in a manner similar to those supplied to an anonymous agent machine. However, after the command has completed, the user's virtual machine is logged off, rather than returned to the anonymous agent pool.

Usage Notes

- All agent virtual machines (RXAGENTn machines) must use the REXEC server's PROFILE EXEC to function properly.
- Because the REXEC server and the agent or user virtual machines communicate using CP messages, the Single Console Image Facility (SCIF) may not be used to monitor the REXEC server, agent, or user consoles.

For more information about using REXEC commands with agent and user machines, and restrictions on their use, see the *TCP/IP User's Guide*, specifically the chapter that deals with using the remote execution protocol.

Chapter 20. Configuring the SMTP Server

The Simple Mail Transfer Protocol (SMTP) server virtual machine can be configured to:

- operate as either an "open" or a *secure* mail gateway between TCP/IP network users, and users located on an RSCS network that is attached to the local host. This allows, for example, OfficeVision users to exchange mail with UNIX users through the VM TCP/IP SMTP gateway.
- direct mail to a *mailer* server virtual machine, or to a specific remote SMTP server for processing.
- process mail using mail aliases, forwarding, and distribution lists defined in an SMTP NAMES file.
- change mail header addresses and specify header address transformations.

To configure the SMTP server, you must perform the following steps:

SMTP Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the system (CP) directory for the SMTP server.
3. Update the DTCPARMS file for the SMTP server.
4. Update the TCPIP DATA file for domain name resolution.
5. Customize the SMTP CONFIG file.
6. Perform advanced SMTP server configuration, if needed.

| **Dynamic System Operation**

| The SMTP server provides a VM Special Message (SMSG) interface that allows you
| to perform server administration tasks through a set of privileged commands. For
| more information, see "Dynamic Server Operation" on page 488.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  SMTP    0
```

The SMTP server requires that port TCP 25 be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
PORT
  25 TCP SMTP  ; SMTP Server
```

Step 2: Update the System (CP) Directory for the SMTP Server

The SMTP virtual machine must be authorized to use the *SPL system service for reading spool files. To use this service, the following CP directory statement must be added for each SMTP server:

```
IUCV *SPL
```

Step 3: Update the DTCPARMS File

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

```
:nick.SMTP
  :Parms.
```

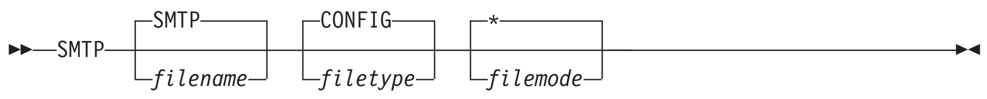
If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the SMTP server if you use a configuration file other than SMTP CONFIG.

SMTP Command

SMTP services are initiated using the SMTP command:



Specify SMTP command operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

filename
The file name of the SMTP server configuration file. The default file name is SMTP.

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

filemode
The file mode of the configuration file. The default file mode is *.

Step 4: Update the TCPIP DATA File for Domain Name Resolution

You can configure the SMTP server to use either a domain name server or the local site tables for host and domain name resolution. To use a domain name server, configure the TCPIP DATA file to define the internet address of one or more name servers using NSINTERADDR statements. For more information, see "Chapter 5. Defining the TCP/IP System Parameters" on page 139. If the TCPIP DATA file does not identify any name servers, local site tables are used. However, if the SMTP server is configured to use name servers, SMTP does not use these site tables.

To obtain detailed information about how the SMTP server resolves a particular host name (for debugging purposes), you can use the SMTP **TRACE RESOLVER** configuration file statement or SMSG command; these are described in more detail on pages 470 and 502.

When SMTP uses a name server, it asks for the MX records for the host to which it is trying to connect. If such MX records are unavailable, the A records for the host are used. For more information about MX records, see "Use of MX Records" or *RFC 974, Mail Routing and the Domain System*.

Step 5: Customize the SMTP CONFIG File

The SMTP configuration file, SMTP CONFIG, defines how the SMTP server is to operate, and what services it provides. See "SMTP Server Configuration File Statements" on page 446 for detailed information about how to specify entries within this file. A sample SMTP configuration file is provided as SMTP SCONFIG on the TCPMAINT 591 disk. Your customized SMTP configuration file should be copied to the TCPMAINT 198 minidisk as SMTP CONFIG. Table 25 on page 446 provides a summary of the configuration statements.

Step 6: Additional SMTP Server Considerations

Before you configure the SMTP server using the statements described in "Step 5: Customize the SMTP CONFIG File" on page 445, you may want to review the information in the following sections:

- "Configuring a TCP/IP-to-RSCS Mail Gateway" on page 473
- "Configuring a TCP/IP-to-RSCS Secure Mail Gateway" on page 475
- "Defining Nicknames and Mailing Lists" on page 478

These sections discuss additional configuration steps and files that are necessary for the SMTP server to run correctly when configured as a mail gateway, or to process mail using mail aliases or forwarding and distribution lists.

The sections that follow provide information about how MX records defined within the Domain Name System and the classification of a mail recipient can affect SMTP server operations.

Use of MX Records

MX records are a type of resource record used in the Domain Name System. For more information about MX and Resource records, see "Resource Records" on page 240. MX records direct the SMTP server to deliver mail to alternative hosts. They are obtained from a domain name server. If SMTP is configured to not use a name server, MX records are not used.

For example, if SMTP wants to send mail to USER@HOST, it checks the name server for MX records and finds the following:

```
HOST    MX    0    HOST
HOST    MX    5    HOST-BACKUP1
HOST    MX    10   HOST-BACKUP2
```

SMTP delivers the mail to the record (host) that has the lowest count; in this example, mail is delivered directly to `HOST`. If `HOST` cannot receive the mail, SMTP tries to deliver it to HOST-BACKUP1. If HOST-BACKUP1 cannot receive the mail, it then tries to deliver it to HOST-BACKUP2. If none of these hosts can receive the mail, SMTP stores the mail and queues it for later delivery, at which time this delivery process is repeated.

If SMTP does not find MX records for a host, it delivers mail to only the primary host.

Local versus Non-local Mail Recipients

When SMTP processes a piece of mail, it determines whether the recipient of that mail is a *local* or a *non-local* user, to allow proper handling of that mail. For example, the result of this determination can affect the mail forwarding processing performed by SMTP.

When a recipient is determined to be a *local* user, the z/VM SMTP server then handles the delivery of that mail itself, possibly through the use of a local mailer, or other network services. SMTP considers a user to be a local user if that user is either:

- a user ID on the local z/VM system
- defined in its SMTP NAMES file, through either a mail alias, a mail forwarding entry, or a mail distribution list
- a user located on an RSCS network that is attached to the local host (applies only to SMTP mail gateway configurations)

If these criteria do not apply to the mail recipient, SMTP considers that user to be a *non-local* user, and will take necessary actions to process that user's mail.

SMTP Server Configuration File Statements

Table 25 provides a summary of the configuration file statements. Keep in mind the following when configuration statements are specified:

- Tokens are delimited by blanks and record boundaries.
- All characters to the right of, and including, a semicolon are treated as comments.

Table 25. SMTP CONFIG Configuration Statements

Statement	Description	Page
ALTRSCSDOMAIN	Specifies an alternative domain name of the RSCS network, if SMTP is running as a mail gateway.	448
ALTTCPHOSTNAME	Specifies an additional host name for the local host. Mail received for this host name is accepted and delivered locally.	448
BADSPOOLFILEID	Specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail.	449
DBCS	Specifies that SMTP should perform DBCS code conversion on mail.	449
FINISHOPEN	Specifies the SMTP wait time for connection.	451
FORWARDMAIL	Specifies whether or not to forward mail to non-local users, or identifies a user exit to control mail forwarding.	451
GATEWAY	Specifies operation of SMTP as a gateway.	452
INACTIVE	Specifies the SMTP wait time before closing an inactive connection.	453
IPMAILERADDRESS	Specifies the address on which SMTP queues mail when it cannot resolve the recipient's address.	453

Table 25. SMTP CONFIG Configuration Statements (continued)

Statement	Description	Page
LOCALFORMAT	Specifies the spool file format for local host mail delivery.	454
LOG	Directs SMTP to log all SMTP traffic.	454
MAILER	Specifies the address of a batch SMTP server to receive mail, in BATCH SMTP format, for various classes of mail recipients.	455
MAILHOPCOUNT	Specifies a limiting value that is used by SMTP to detect a mail delivery loop condition and then cease delivery attempts for a mail item.	455
MAXMAILBYTES	Specifies the maximum size of mail that is accepted over TCP/IP connections.	456
NOLOG	Turns off logging of mail transactions.	457
ONDISKFULL	Specifies a set of CP commands for SMTP to execute when specified SMTP 191 disk-full thresholds are crossed.	457
OUTBOUNDOPENLIMIT	Specifies a limit on the maximum number of simultaneous TCP/IP connections over which SMTP actively delivers mail.	458
PORT	Specifies an alternative port number for SMTP server during testing.	458
POSTMASTER	Specifies the address (or addresses) for mail addressed to the postmaster at the local host.	459
RCPTRESPONSEDELAY	Specifies the amount of time the SMTP server delays responding to the RCPT commands.	460
RESOLVERRETRYINT	Specifies the number of minutes SMTP waits between attempts to resolve domain names.	460
RESTRICT	Specifies addresses of users who are not allowed to use SMTP mail services.	461
RETRYAGE	Specifies the number of days after which mail is returned as undeliverable.	461
RETRYINT	Specifies the number of minutes between attempts to send mail to an inactive TCP/IP host.	462
REWRITE822HEADER	Specifies whether SMTP should rewrite the RFC 822 headers of mail arriving from the RSCS side of the mail gateway.	462
RSCSDOMAIN	Specifies the domain name of the RSCS network (if SMTP is running as a mail gateway).	463
RSCSFORMAT	Specifies the spool file format for mail delivered to recipients on the RSCS network.	464
SECURE	Specifies that SMTP operates as a secure mail gateway between TCP/IP network sites and RSCS network sites.	464
SMSGAUTHLIST	Specifies the addresses of users authorized to issue privileged SMTP SMSG commands.	464
SMTPCMDS	Defines the characteristics of the SMTP command user exit and indicates whether or not the exit is to be enabled (on) or disabled (off) when the server completes initialization.	465
SOURCEROUTES	Specifies whether or not SMTP will honor source routes.	467
SUPRESSNOTIFICATION	Specifies that SMTP should not send a notification to the mail sender when mail delivery is successful.	469
TEMPERRORRETRIES	Specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem.	469
TRACE	Specifies which type of tracing should be enabled during server initialization.	469

Table 25. SMTP CONFIG Configuration Statements (continued)

Statement	Description	Page
VERIFYCLIENT	Specifies whether or not client verification is to be performed. Client verification can be performed using the built-in client verification function or through a user exit.	470
VERIFYCLIENTDELAY	Specifies the amount of time (in seconds) that SMTP will wait for the domain name system to respond during client verification.	471
WARNINGAGE	Specifies the number of days after which a copy of the mail is returned to the sender.	472
8BITMIME	Specifies the file name of the translation table to be used for 8-bit MIME support.	472

ALTRSCSDOMAIN Statement

The ALTRSCSDOMAIN statement specifies an alternate domain name for an RSCS network (if SMTP is configured to operate as a mail gateway). This alternate domain name is used in the same manner, but in addition to any domain name defined using the RSCSDOMAIN statement. Only one ALTRSCSDOMAIN statement can be specified within the SMTP configuration file.

```
►►──ALTRSCSDOMAIN──domain──────────────────────────────────────────►◄
```

Operands

domain
The alternate domain name of the RSCS network. The alternative RSCS domain name is a string of 1 to 64 alphanumeric characters.

ALTTCPHOSTNAME Statement

The ALTTCPHOSTNAME statement specifies an alternate fully qualified host name by which SMTP knows the local host. Mail sent to users at *hostname* are treated as if they were local users.

```
►►──ALTTCPHOSTNAME──hostname───────────────────────────────────────►◄
```

Operands

hostname
The alternate TCP/IP host name of this host.

BADSPOOLFILEID Statement

The BADSPOOLFILEID statement specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail. You can specify the BADSPOOLFILEID statement only once.

```
     ┌─BADSPOOLFILEID TCPMAINT──┐
►►───┤                          ├──────────────────────────────────────────►◄
     └─BADSPOOLFILEID user_id───┘
```

Operands

user_id
: The user ID on the local system to which bad spool files and looping mail are delivered; *user_id* should be a maximum of 8 characters. The default is TCPMAINT.

DBCS Statement

The DBCS statement specifies that SMTP is to perform DBCS code conversion on processed mail. The parameters of the DBCS statement determine which translation table should be used in the conversion.

The following example shows the format of the DBCS statement.

```
►►──DBCS──┬──┬─JIS78KJ─┬───┬─ASCII────┬─┬──────────────────────────────►◄
          │  └─JIS83KJ─┘   └─JISROMAN─┘ │
          ├─SJISKANJI───────────────────┤
          ├─EUCKANJI────────────────────┤
          ├─IBMKANJI────────────────────┤
          ├─HANGEUL─────────────────────┤
          ├─KSC5601─────────────────────┤
          └─TCHINESE────────────────────┘
```

Operands

JIS78KJ
: Indicates IBM Kanji to JIS 1978 Kanji conversion is to be performed. The JIS 1978 Shift-Out codes are `ESC`, `$`, and `@`. SMTP will load the JIS78KJ DBCS translation table from the TCPKJBIN binary translate table file.

JIS83KJ
: Indicates IBM Kanji to JIS 1983 Kanji conversion is to be performed. The JIS 1983 Shift-Out codes are `ESC`, `$`, and `B`. SMTP will load the JIS83KJ DBCS translation table from the TCPKJBIN binary translate table file.

ASCII
: Indicates mail is to be shifted in ASCII code from JIS Kanji code. The ASCII Shift-In codes are `ESC`, `(`, and `B`.

JISROMAN
Indicates mail is to be shifted in JISRoman code from JIS Kanji code. The JISRoman Shift-In codes are `ESC`, `(`, and `J`.

SJISKANJI
Indicates IBM Kanji to Shift-JIS Kanji conversion is to be performed. SMTP will load the SJISKANJI DBCS translation table from the TCPKJBIN binary translate table file.

EUCKANJI
Indicates IBM Kanji to Extended UNIX Code Kanji conversion is to be performed. SMTP will load the EUCKANJI DBCS translation table from the TCPKJBIN binary translate table file.

IBMKANJI
Indicates IBM (EBCDIC) Kanji conversion is to be used. This option causes **no** conversion to be performed on the body of the mail; it may be used for sending and receiving mail in EBCDIC. If this option is selected, other SMTP servers on the same network should all be configured to perform IBMKANJI conversions. If IBMKANJI is specified, and LOCALFORMAT or RSCSFORMAT is set to PUNCH, then mail received in ASCII may be folded to inconsistent record lengths. LOCALFORMAT and RSCSFORMAT should be set to NETDATA in this case.

Indicates IBMKANJI transfer type does not require any translate table to be loaded.

HANGEUL
Indicates IBM Hangeul to Hangeul PC code conversion is to be performed. SMTP will load the HANGEUL DBCS translation table from the TCPHGBIN binary translate table file.

KSC5601
Indicates IBM Hangeul to KSC-5601 PC code conversion is to be performed. SMTP will load the KSC5601 DBCS translation table from the TCPHGBIN binary translate table file.

TCHINESE
Indicates IBM Traditional Chinese to Traditional Chinese 5550 PC code conversion is to be performed. SMTP will load the TCHINESE (5550) DBCS translation table from the TCPCHBIN binary translate table file.

The SBCS translation table, STANDARD TCPXLBIN, will be used to convert mail headers, with the body of the mail being converted using a DBCS translation table from SMTP TCPKJBIN, SMTP TCPHGBIN, or SMTP TCPCHBIN. If SMTP TCPKJBIN, SMTP TCPHGBIN, or SMTP TCPCHBIN, cannot be found, then SMTP will use STANDARD TCPKJBIN, STANDARD TCPHGBIN, or STANDARD TCPCHBIN.

DBCS conversion is performed only on outgoing and incoming mail to and from TCP/IP-connected hosts. Mail spooled to SMTP for the local host is delivered directly, without any DBCS code conversion.

Note: For more information on loading and customizing DBCS translate tables, see "Customizing DBCS Translation Tables" on page 601.

FINISHOPEN Statement

The FINISHOPEN statement specifies the number of seconds SMTP waits while trying to establish a connection to a foreign site. After the specified number of seconds, SMTP aborts the connection.

```
     ┌─FINISHOPEN 120─────┐
►►───┼────────────────────┼──────────────────────────────────────────►◄
     └─FINISHOPEN seconds─┘
```

Operands

seconds
: An integer in the range of 1 through 86,400 (24 hours) indicating the number of seconds to wait for a connection to open. The default FINISHOPEN time-out is 120 seconds.

FORWARDMAIL Statement

The FORWARDMAIL statement is used to indicate whether or not to forward mail to non-local users, or to identify a user exit to be used to control mail forwarding. For information on using the mail forwarding exit, see the *TCP/IP Programmer's Reference*.

```
     ┌─FORWARDMAIL YES ENDFORWARDMAIL────────────────────────────────┐
►►───┼───────────────────────────────────────────────────────────────┼────►◄
     └─FORWARDMAIL──┬─NO───────────────────────────────┬─ENDFORWARDMAIL─┘
                    │      ┌─SMTPFWDX EXEC──┐  ┌─ON──┐ │
                    └─EXIT─┼────────────────┼──┼─────┼─┘
                           │         ┌─EXEC─┐ │  └─OFF─┘
                           └─exitname┼──────┼─┘
                                     └─TEXT─┘
```

Operands

YES
: Indicates mail forwarding is to be performed; this is the default.

NO
: Indicates that no mail forwarding is to be performed. When SMTP determines that the recipient is not on the local system, the RCPT TO: command will be rejected.

EXIT
: Indicates a mail forwarding exit routine is being defined.

exitname
: The name of the exit routine associated with this command; the default exit name is **SMTPFWDX**.

EXEC
: Indicates the exit routine name specified on this command is the name of an EXEC; this is the default.

TEXT
Indicates the exit routine name specified on this command is the name of a text deck.

ON
Indicates the specified exit (being defined with this command) is to be enabled (turned on).

OFF
Indicates the specified exit (being defined with this command) is to be disabled (turned off).

Examples

- The following SMTP configuration file entry will enable the mail forwarding exit routine SMTPFWDX EXEC.

  ```
  ForwardMail
     exit smtpfwdx exec on
  EndForwardMail
  ```

  When this entry is processed, the following text is displayed during server initialization:

  ```
  Forward Mail                  : Exit SMTPFWDX EXEC ON
  ```

- This next entry defines the mail forwarding exit routine SMTPFWDX TEXT, but disables its use once SMTP server initialization is complete. Thus, mail forwarding will be allowed and performed.

  ```
  ForwardMail
     exit smtpfwdx text off
  EndForwardMail
  ```

  When this entry is processed, the following text is displayed during server initialization:

  ```
  Forward Mail                  : Yes (Exit SMTPFWDX TEXT OFF)
  ```

GATEWAY Statement

The GATEWAY statement specifies that SMTP is to operate as a mail gateway between TCP/IP network sites and RSCS network sites (provided the host system is connected to both a TCP/IP network and an RSCS network).

If GATEWAY is specified, SMTP accepts mail addressed to users on RSCS hosts defined in the SMTPRSCS HOSTINFO file. For more information about setting up a gateway node, see "Configuring a TCP/IP-to-RSCS Mail Gateway" on page 473. If GATEWAY is not specified, SMTP rejects all mail that arrives from RSCS.

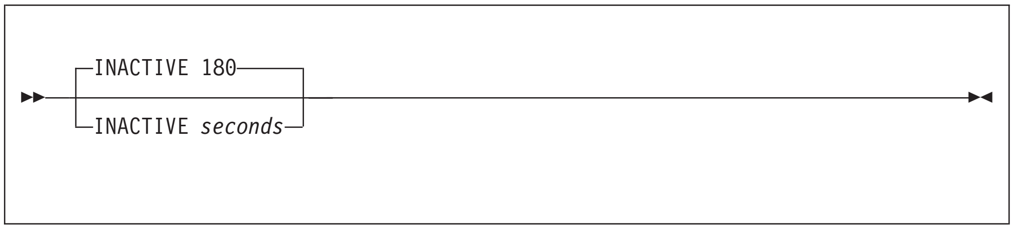
```
►►──GATEWAY──────────────────────────────────────────────────────►◄
```

Operands

The GATEWAY statement has no operands.

INACTIVE Statement

The INACTIVE statement specifies the period of inactivity (in seconds) after which SMTP considers a connection to be broken (that is, unusable). After the specified amount of inactive time, SMTP closes the connection.



Operands

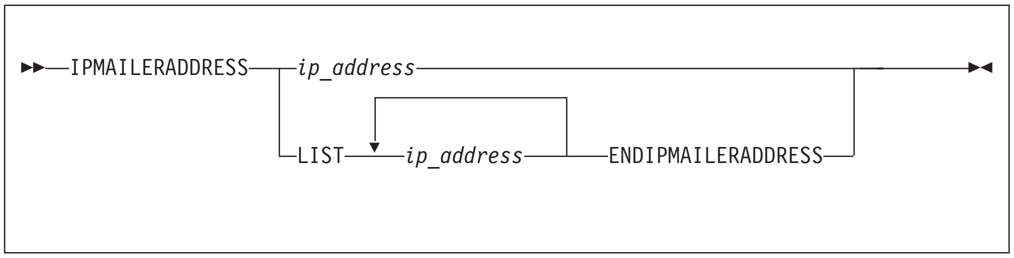
seconds
: An integer, in the range of 1 through 86,400 (24 hours), that specifies the number of seconds after which SMTP considers a connection to be broken. The default INACTIVE time-out period is 180 seconds.

IPMAILERADDRESS Statement

The IPMAILERADDRESS statement specifies an IP address to which SMTP queues mail when it cannot resolve a mail recipient address.

Note: The IPMAILERADDRESS statement and the UNKNOWN parameter of the MAILER statement are mutually exclusive. The SMTP server will not initialize if both are specified in the SMTP configuration file.

IPMAILERADDRESS is useful when your VM system does not have full connectivity to or knowledge of other networks. It allows you to direct mail to a different SMTP server that may be able to deliver mail to requested hosts that reside on these other networks.



Operands

ip_address
: The dotted-decimal internet address of the host to which SMTP is to direct mail when it cannot resolve a mail recipient address.

LIST
: Indicates that a list of IP addresses will be given to which SMTP will direct a piece of mail when it cannot resolve a mail recipient address. SMTP will attempt to direct the mail to the first IP address in the list. If unable to connect to that address, SMTP will proceed through the list sequentially until the mail is redirected or the list is exhausted.

LOCALFORMAT Statement

The LOCALFORMAT statement specifies the spool file format for mail delivered to recipients on the local host. The default format is NETDATA.

For PUNCH format:
- The spool file is in CMS PUNCH NOHEADER format.
- Records are folded to 80 or fewer characters in length.
- Spool file class M is used.
- The file name is the first 8 characters of the user ID of the sender.
- The file type is MAIL.

For NETDATA format:
- The spool file is in NETDATA format.
- Records can be longer than 80 characters.
- Spool file class A is used.
- The file name is the first 8 characters of the user ID of the sender.
- The file type is NOTE.

```
    ┌─LOCALFORMAT NETDATA─┐
►►──┤                     ├──────────────────────────────────────────►◄
    └─LOCALFORMAT PUNCH───┘
```

Operands

PUNCH
Indicates that records are folded to 80 or fewer characters in length.

NETDATA
Indicates that records can be longer than 80 characters and arrive as MESSAGE-type records. The default format is NETDATA.

LOG Statement

The LOG statement specifies that SMTP is to log information about all SMTP traffic. The origin, sender, and recipients of each piece of mail are written to the log as mail is received and delivered. To turn off logging, use the NOLOG statement.

```
    ┌─LOG DISK──┐
►►──┤           ├────────────────────────────────────────────────────►◄
    └─LOG SPOOL─┘
```

Operands

DISK
: Indicates that log information is to be appended to the SMTP log file; DISK is the default.

SPOOL
: Indicates that log information is to be written to the console.

MAILHOPCOUNT Statement

The MAILHOPCOUNT statement is used by SMTP to detect a mail delivery loop condition between itself and another mail server. When this condition arises, SMTP ceases delivery attempts for a mail item.

```
                  ┌─25────────┐
►►──MAILHOPCOUNT──┴─rcv_limit─┴───────────────────────────────────────►◄
```

Operands

rcv_limit
: An integer in the range of 5 to 300 that specifies the maximum number of `Received` lines that can exist within the header section of a given mail item. The default is 25.

  When the `Received` lines present the header section of a given mail item matches the specified *rcv_limit* value, SMTP considers the delivery process for that piece of mail to be in a loop and stops its attempts to deliver that mail item. The mail item in question (and its corresponding ADDRFILE file) are renamed and saved by SMPT at file mode A.

MAILER Statement

The MAILER statement identifies a virtual machine to receive mail, in BATCH SMTP (BSMTP) format, for various classes of mail recipients. The MAILER virtual machine must exist on either the local node or a node on the RSCS network. This virtual machine must accept batch SMTP format spool files and deliver the mail or return undeliverable mail.

```
►►──MAILER──┬─user_id─────────┬──┬─PUNCH───┬──┬─SOURCEROUTES───┬──────────►
            └─user_id@node_id─┘  └─NETDATA─┘  └─NOSOURCEROUTES─┘

►──┬─LOCAL───┬──┬─RSCS───┬──┬─UNKNOWN───┬────────────────────────────────►◄
   └─NOLOCAL─┘  └─NORSCS─┘  └─NOUNKNOWN─┘
```

Operands

user_id
: The user ID of a MAILER virtual machine; this machine is presumed to reside on the local RSCS node.

user_id@node_id
: The user ID and RSCS node of a MAILER virtual machine.

PUNCH
Indicates that the remote MAILER virtual machine can accept only PUNCH format spool files. BSMTP header records longer than 80 characters are split, and an EBCDIC newline character (X'15') is placed in column 80 to indicate that the record is continued. Records within the body of the mail longer than 80 characters are split across multiple PUNCH records.

NETDATA
Indicates that the MAILER virtual machine accepts NETDATA format spool files. The NETDATA protocol automatically handles records longer than 80 characters.

SOURCEROUTES
Indicates that the MAILER virtual machine accepts BSMTP header addresses with source routes.

NOSOURCEROUTES
Indicates that the MAILER virtual machine does not accept source routes in the BSMTP header addresses.

LOCAL
Indicates that mail for local recipients is spooled to the MAILER virtual machine.

NOLOCAL
Indicates that mail for local recipients is spooled directly to the recipients.

RSCS
Indicates that mail for recipients on the RSCS network is spooled to the MAILER virtual machine.

NORSCS
Indicates that mail for recipients on the RSCS network is spooled directly to the recipients.

UNKNOWN
Indicates that mail for recipients on an unknown host is spooled to the MAILER virtual machine.

NOUNKNOWN
Indicates that mail for recipients on unknown hosts is returned to the sender as undeliverable.

MAXMAILBYTES Statement

The MAXMAILBYTES statement specifies the maximum size, in bytes, of mail that is accepted over a TCP connection.

```
   ┌─MAXMAILBYTES 524288─┐
►►─┤                     ├─────────────────────────────────────────────►◄
   └─MAXMAILBYTES bytes──┘
```

Operands

bytes
The maximum number of bytes to accept. Mail larger than this size that arrives

over a TCP connection is rejected. The limits for this statement are 1 and 2,147,483,647. The default byte size is 524,288.

NOLOG Statement

The NOLOG statement turns off the logging of information about SMTP traffic. For more information, see “LOG Statement” on page 454.

```
►►──NOLOG──────────────────────────────────────────────────────────►◄
```

Operands

The NOLOG statement has no operands.

ONDISKFULL Statement

The ONDISKFULL statement specifies a set of CP commands (such as CP MSG or CP SMSG) for SMTP to execute when specified SMTP 191 disk-full thresholds are crossed.

```
                  ┌───────────┐         ┌─────────────────┐
                  ▼           │         ▼                 │
►►──ONDISKFULL────threshold───┴──ACTIONS────BEGINactionEND──┴──ENDACTIONS───►◄
```

Operands

threshold
: The disk-full percentages for which actions should be performed. Positive numbers cause the actions to execute when the SMTP 191 minidisk percentage exceeds the percent specified. Negative numbers cause the actions to execute when the minidisk utilization drops below the percent specified.

action
: A CP command that is to be issued when a threshold condition is crossed. To simplify the issuing of informational messages, the special keyword ***MESSAGE*** can be included as part of the `action` string. This will cause one of the following text strings (as warranted) to be substituted in place of `*MESSAGE*`:

```
date time SMTPserver_id at SMTPnode_id - Disk Above nn Percent Full
```

or

```
date time SMTPserver_id at SMTPnode_id - Disk Below nn Percent Full
```

There is no limit on the number of BEGIN *action* END statements that you can specify. All commands are converted to uppercase, and extra blanks are removed. The CP command output for all `action` commands is suppressed, although nonzero command return codes will be reported. The ENDACTIONS statement ends the ONDISKFULL statement.

In the following example, the ONDISKFULL statement causes messages to be sent to two users (MATT at ENDICOTT and TCPMAINT on the local system) when the SMTP 191 minidisk exceeds 40, 50, 60, 70, 80, and 90 percent full. Messages are also sent when the disk full percentage decreases below these same thresholds.

```
;
;  On Disk Full Actions
;
ONDISKFULL 40 50 60 70 80 90 -90 -80 -70 -60 -50 -40 ACTIONS
BEGIN CP SMSG RSCS MSG ENDICOTT MATT *MESSAGE* END
BEGIN CP MSG TCPMAINT *MESSAGE* END
ENDACTIONS
;
```

For the above statements, the message that follows would be received by user MATT at ENDICOTT when the 191 minidisk of the SMTP server at WATSON exceeds 50 percent full:

```
From WATSON(SMTP): 03/17/99 21:54 SMTP at WATSON - Disk Above 50 Percent Full
```

whereas the TCPMAINT user (on the local WATSON system) would receive this message:

```
03/17/99 21:54 SMTP at WATSON - Disk Above 50 Percent Full
```

OUTBOUNDOPENLIMIT Statement

The OUTBOUNDOPENLIMIT statement specifies the limit on the maximum number of simultaneous TCP connections over which SMTP can actively deliver mail. By default, SMTP operates with no limits, and opens as many TCP connections as necessary to ensure the fastest possible delivery of mail. The OUTBOUNDOPENLIMIT statement should be specified only if there are limited TCP resources on the system and SMTP is using an unfair portion of these resources.

```
►►──OUTBOUNDOPENLIMIT──connections─────────────────────────────────────►◄
```

Operands

connections
A number in the range of 1 to the maximum number of TCP connections available on your system. If *connections* is out of range, no limit is imposed.

PORT Statement

The PORT statement causes SMTP to listen on a specified TCP connection port. By convention, port number 25 is usually reserved (in the TCPIP server configuration file) for the SMTP server to accept incoming mail requests.

```
      ┌─PORT 25──────────┐
►►────┤                  ├──────────────────────────────────────────►◄
      └─PORT port_number─┘
```

Operands

port_number
: An integer in the range of 1 through 65,535 that specifies the port number to which SMTP listens. The default is port 25.

POSTMASTER Statement

The POSTMASTER statement identifies the user that should receive mail sent to the “postmaster” of the system where the SMTP server runs. A postmaster is generally responsible for managing the mail system for a site. As such, other users in the internet community might send mail to the postmaster, to report mail problems they believe are associated with that site, or to have mail routed to a recipient (either local or in a different domain) for which the correct address is not known.

```
      ┌─POSTMASTER TCPMAINT──────────────┐
►►────┤                                  ├──────────────────────────►◄
      └─POSTMASTER──┬─user_id──────────┬─┘
                    └─user_id@hostname─┘
```

Operands

user_id
: A user ID on the local system to which mail addressed to “postmaster” should be delivered; the default is TCPMAINT.

user_id@hostname
: A user ID and host name to which mail addressed to “postmaster” should be delivered. The supplied *user_id* and *hostname* values may identify either a user ID on a specific RSCS node, or an internet user and host name.

Notes:

1. If a POSTMASTER statement is not specified, the default value, TCPMAINT, will be used.
2. If SMTP is *not* operating as a secure mail gateway between TCP/IP and RSCS network sites, multiple POSTMASTER statements can be specified. When this is done, code a separate POSTMASTER statement for each user that should receive mail addressed to “postmaster”. No limit is imposed on the number of POSTMASTER statements that can be coded.
3. If SMTP is operating as a secure mail gateway between TCP/IP and RSCS network sites, only one POSTMASTER statement can be specified. Additionally, *user_id* must be a user ID defined for the local system.

RCPTRESPONSEDELAY Statement

The RCPTRESPONSEDELAY statement specifies the amount of time the SMTP server delays its response to a RCPT TO: command (supplied by a remote STMP “sender” host) while host domain resolution of a mail recipient is being performed. If SMTP does not receive a domain name server (DNS) response within the specified period, it *assumes* the host resolution is successful and does the following:

- Sends a `250 OK` response to the “sender” SMTP host
- Queues the mail locally, so that asynchronous resolution of the mail recipient can be performed.

If SMTP later determines that the recipient address cannot be resolved, the queued mail is returned to the sender.

```
    ┌─RCPTRESPONSEDELAY 60──────┐
►►──┼───────────────────────────┼────────────────────────────────────────►◄
    └─RCPTRESPONSEDELAY seconds─┘
```

Note: When the client sends multiple commands (including the RCPT command) without waiting for the response from the SMTP server, the SMTP server responds immediately to the RCPT command with the 250 OK message. The RCPTRESPONSEDELAY value will be ignored.

Operands

seconds
: The number of seconds SMTP waits before responding to the RCPT TO: command. The range is 0 through 86,400 (24 hours). The default is 60 seconds.

RESOLVERRETRYINT Statement

The RESOLVERRETRYINT statement specifies the interval (in minutes) the SMTP server should wait between attempts to resolve a host domain name.

```
    ┌─RESOLVERRETRYINT 20──────┐
►►──┼──────────────────────────┼─────────────────────────────────────────►◄
    └─RESOLVERRETRYINT minutes─┘
```

Operands

minutes
: An integer in the range of 1 to 1439 (24 hours - 1 minute) that specifies the number of minutes SMTP should wait between successive attempts to resolve a host domain name. The default is 20 minutes.

RESTRICT Statement

The RESTRICT statement identifies users who may not send mail through this SMTP server. If SMTP receives a spool file from a restricted user, the spool file is purged, returned to the sender or transferred to a third party, depending on the options specified.

In addition, SMTP rejects any MAIL FROM: or RCPT TO: commands whose destinations are restricted users.

SMTP rejects only addresses that are in the restrict list; it does not check for aliases. For example, you can restrict `user@host1`. If `host2` is an alias for `host1`, mail for `user@host2` is not rejected, unless `user@host2` is also in the restrict list.

```
                                                  ┌──────────────────┐
                                                  ▼                  │
►►──RESTRICT──┬─PURGE──────────────────────┬────user_id@node_id──┴──────────────►
              ├─RETURN─────────────────────┤
              └─TRANSFERTO transfer_id─────┘
►─ENDRESTRICT────────────────────────────────────────────────────────────────────►◄
```

Note: The RESTRICT statement cannot be used if the SMTP virtual machine is running as a secure gateway. Either remove or comment out the RESTRICT statements from the SMTP CONFIG file.

Operands

PURGE
Indicates that spool files from restricted users are to be purged.

RETURN
Indicates that spool files from restricted users are to be returned to each respective user.

TRANSFERTO
Indicates that spool files from restricted users are to be transferred to a specific user ID.

transfer_id
The user ID to which the spool file is transferred.

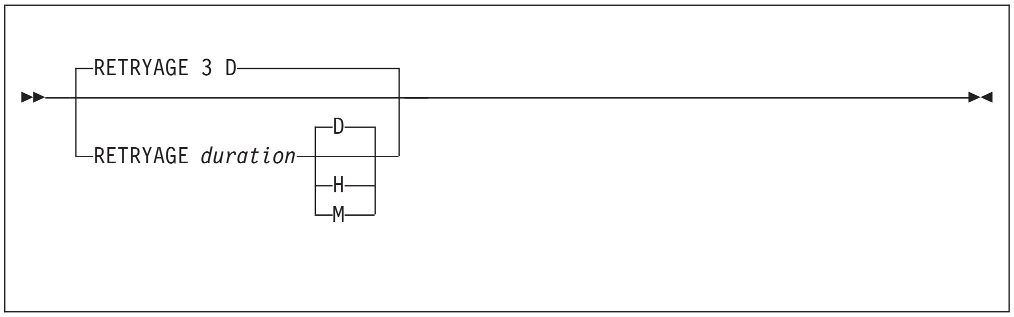
user_id@node_id
The user ID and node ID of a restricted user. This parameter can be repeated, and can include an asterisk (*) to act as a wildcard.

The ENDRESTRICT statement ends the RESTRICT statement.

RETRYAGE Statement

The RETRYAGE statement specifies the amount of time after which SMTP returns mail as undeliverable. SMTP tries to deliver mail to an inactive site at intervals determined by the RETRYINT statement. After the amount of time specified by the

RETRYAGE statement has passed, SMTP returns the mail to the sender with a note that lists the recipients to which the mail could not be delivered.



Operands

duration
: The amount of time (specified in days, hours, or minutes) over which SMTP is to attempt delivery of a piece of mail. If *duration* is specified in days, the range is 0 through 365; when it is given in hours, the range is 0 through 8760 (365 days). When *duration* is specified in minutes, the range is 0 through 525600 (365 days). The default is 3 days.

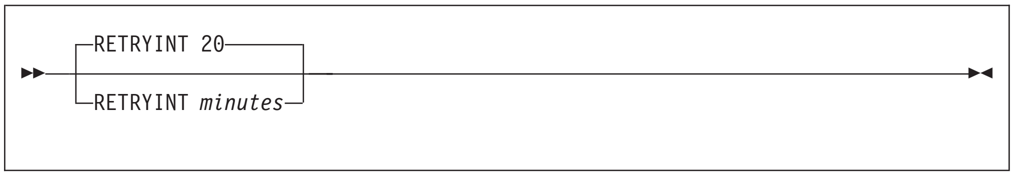
D Indicates that *duration* is specified in days. This is the default.

H Indicates that *duration* is specified hours.

M Indicates that *duration* is specified in minutes.

RETRYINT Statement

The RETRYINT statement specifies the number of minutes SMTP should wait between attempts to deliver mail to a TCP/IP host that is inactive (that is, not responsive to connection attempts).



Operands

minutes
: An integer in the range of 0 to 1440 (24 hours) that specifies the number of minutes SMTP should wait between mail delivery attempts to an inactive host. The default is 20 minutes.

REWRITE822HEADER Statement

The REWRITE822HEADER statement specifies the direction to SMTP to rewrite or print the RFC 822 headers of mail arriving from the RSCS side of the mail gateway.

```
       ┌─REWRITE822HEADER YES NOPRINT────────────┐
►►─────┤                                         ├──────────────────────────────────►◄
       └─REWRITE822HEADER──┬─YES──┬─PRINT───┬─┬─┘
                           │      └─NOPRINT─┘ │
                           └─NO───────────────┘
```

Operands

YES
Indicates that SMTP should rewrite the RFC 822 mail headers. The YES parameter with NOPRINT is the default. SMTP uses a set of default header rewriting rules. For more information to customize the rewriting rules, see "Customizing SMTP Mail Headers" on page 479.

NO
Indicates that SMTP should not rewrite the RFC 822 mail headers. This is not recommended unless all mail user agents sending mail to SMTP create RFC 822 mail headers with fully qualified domain addresses that are valid on the internet.

PRINT
Indicates that SMTP should print the RFC 822 header rewriting rules to the console.

NOPRINT
Indicates that SMTP should not print the RFC 822 header rewriting rules to the console.

RSCSDOMAIN Statement

The RSCSDOMAIN statement specifies the domain name of the RSCS network (if SMTP is running as a mail gateway). The RSCS domain name is used to rewrite the header fields of mail passing from RSCS network senders to TCP/IP network recipients.

If TCP/IP network senders qualify an RSCS network recipient's host name with a domain name that matches either the RSCS domain name (defined using this statement) or an alternate RSCS domain name (defined by the ALTRSCSDOMAIN statement), SMTP accepts and attempts local delivery of the supplied mail, due to use of the GATEWAY configuration statement and presence of SMTPRSCS HOSTINFO data. If this delivery attempt fails, the mail is rejected due to an unknown host condition.

If TCP/IP network senders do not qualify the network recipient's host name with an RSCS domain name, SMTP attempts to resolve the given host via DNS services. If a resolved destination cannot be obtained, SMTP then attempts local delivery, as described in the previous paragraph.

Note: SMTP considers the RSCS domain name BITNET to be a synonym for NETNORTH, EARN, and EARNET.

```
►►──RSCSDOMAIN──RSCS_domain_name──────────────────────────────────────────►◄
```

Operands

RSCS_domain_name
A domain name, specified as a string of alphanumeric characters. For example, `BITNET`, `VNET`, and `vnet.ibm.com` are all valid RSCS domain names. The default RSCS domain name is a null string.

RSCSFORMAT Statement

The RSCSFORMAT statement specifies the spool file format for mail delivered to recipients on the RSCS network. This statement is valid only in GATEWAY mode.

```
     ┌─RSCSFORMAT NETDATA─┐
►►───┼────────────────────┼──────────────────────────────────────────────►◄
     └─RSCSFORMAT PUNCH───┘
```

Operands

PUNCH
Indicates that records are folded up to 80 characters in length or fewer.

NETDATA
Indicates that records can be longer than 80 characters and arrive as MESSAGE-type records. The default format is NETDATA.

SECURE Statement

The SECURE statement specifies SMTP operates as a secure mail gateway between TCP/IP network sites and RSCS network sites. Mail is accepted by RSCS only if the user ID and node ID are included in the SMTP SECTABLE file. See "Configuring a TCP/IP-to-RSCS Secure Mail Gateway" on page 475 for more information.

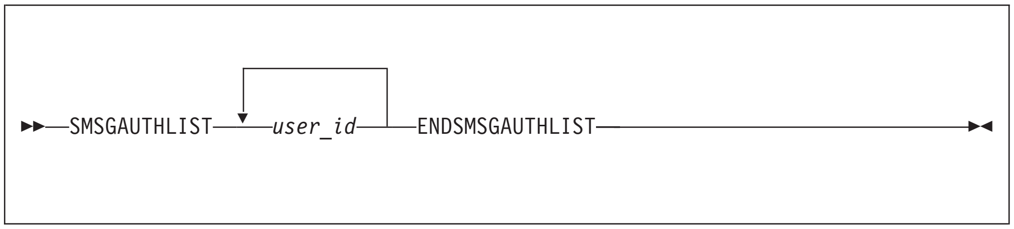
```
►►──SECURE────────────────────────────────────────────────────────────────►◄
```

Operands

The SECURE statement has no operands.

SMSGAUTHLIST Statement

The SMSGAUTHLIST statement identifies the local and RSCS users authorized to issue privileged SMTP SMSG commands. Any VM user can issue the general usage SMTP SMSG commands, but only those users specified in the SMSGAUTHLIST statement can issue the privileged commands. Privileged SMTP SMSG commands allow the shutting down or rebooting of SMTP, the enabling or disabling of various SMTP trace/debug options, and the closing of SMTP's console.



Operands

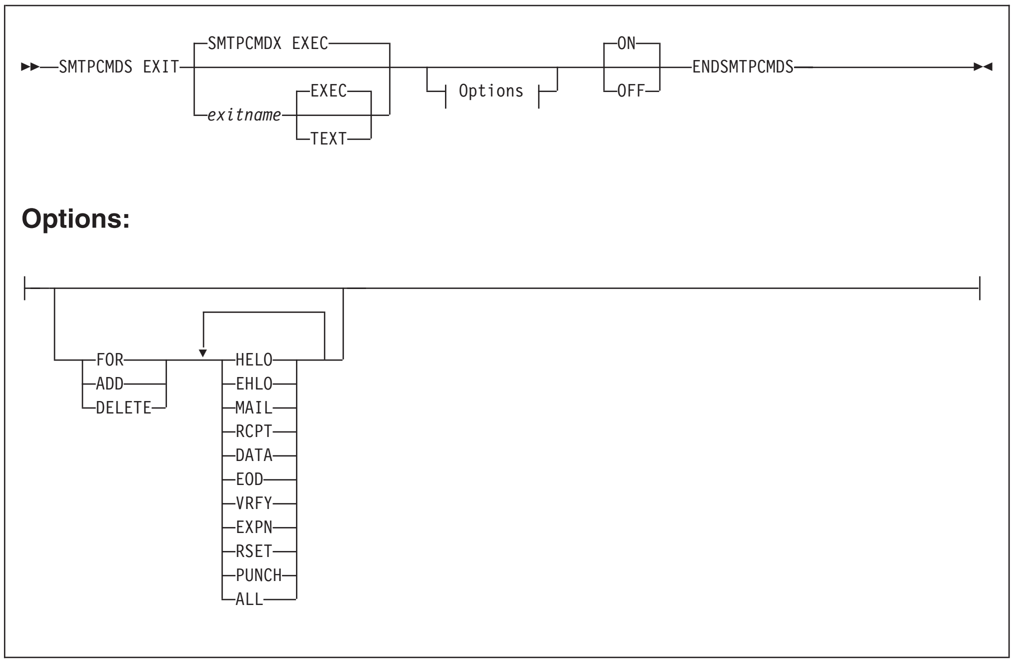
user_id
: The address of a local user ID authorized to issue privileged SMTP SMSG commands.

The *user_id* parameter can be repeated. The ENDSMSGAUTHLIST statement ends the SMSGAUTHLIST statement.

For more information about the SMSG interface to SMTP, see the "Privileged User SMSG Commands" on page 488.

SMTPCMDS Statement

The SMTPCMDS statement is used to define the characteristics of the SMTP command user exit and to indicate whether or not the exit is to be enabled (turned on) or disabled (turned off) when the server completes initialization. For information on using the SMTP command exit, see the *TCP/IP Programmer's Reference*.



Operands

exitname
The name of the exit routine associated with this command; the default exit name is **SMTPCMDX**.

EXEC
Indicates the exit routine name specified on this command is the name of an EXEC; this is the default.

TEXT
Indicates the exit routine name specified on this command is the name of a text deck.

FOR
Precedes the exact list of commands for which the exit is being defined.

ADD
Precedes the list of commands that are to be added to this exit definition.

DELETE
Precedes the list of commands that are to be deleted from this exit definition.

HELO
Indicates the exit routine is to be turned on or off for the HELO command.

EHLO
Indicates the exit routine is to be turned on or off for the EHLO command.

MAIL
Indicates the exit routine is to be turned on or off for the MAIL FROM: command.

RCPT
Indicates the exit routine is to be turned on or off for the RCPT TO: command.

DATA
Indicates the exit routine is to be turned on or off for the DATA command.

EOD
Indicates the exit routine is to be turned on or off when the “end of data” condition is reached; that is, when a period (.) is received by the SMTP server after a DATA command.

VRFY
Indicates the exit routine is to be turned on or off for the VRFY command.

EXPN
Indicates the exit routine is to be turned on or off for the EXPN command.

RSET
Indicates the exit routine is to be turned on or off for the RSET command.

PUNCH
Indicates the exit routine is to be turned on or off for PUNCH processing. If the exit is enabled for this condition, it will be called when SMTP is ready to punch local mail (mail on the same node or RSCS network) to its destination.

ALL
Indicates the exit routine is to be turned on or off for all SMTP commands for which user exit capability is provided.

ON
Indicates the specified exit is being enabled (turned on) for a particular command or set of commands.

OFF
Indicates the specified exit is being disabled (turned off) for a particular command or set of commands.

Examples

- The following SMTP configuration file entry will enable the SMTP command exit routine SMTPCMDX EXEC; exit processing will be performed for only the VRFY and EXPN commands.

```
SmtpCmds
   exit smtpcmdx exec for vrfy expn on
EndSmtpCmds
```

  When this entry is processed, the following text is displayed during server initialization:

```
SMTP Command Exit              : SMTPCMDX EXEC ON
 - defined for                 : VRFY EXPN
```

- This next entry defines the SMTP command exit routine MYEXIT TEXT for only the HELO command, but disables its use once SMTP server initialization is complete.

```
SmtpCmds
   exit myexit text for helo off
EndSmtpCmds
```

  When this entry is processed, the following text is displayed during server initialization:

```
SMTP Command Exit              : MYEXIT TEXT OFF
 - defined for                 : HELO
```

SOURCEROUTES Statement

The SOURCEROUTES statement specifies whether the SMTP server will honor source routes. Source routes may be present on the MAIL FROM: or RCPT TO: commands processed by SMTP. With this statement, you may specify which source routes are honored.

A source route is a path that contains a routing list of hosts and a destination mailbox. The list of hosts is the *route* — information about how the mail is to arrive at its final destination; the mail is passed from one host in this list to the next until it is delivered to the intended recipient.

The specification that follows is an example of a **source route**:

```
<@HOST1,@HOST2,@HOST3:USER@HOST4>
```

The list of hosts is HOST1, HOST2 and HOST3, and the destination is USER@HOST4.

If this sample source route is included with a RCPT TO: command, and is honored by SMTP, the mail will be sent to HOST1, then to HOST2, then to HOST3 and finally to USER@HOST4. When source routes are not honored, mail is sent directly to USER@HOST4; the list of hosts is ignored.

If this sample source route is included with a MAIL FROM: command and source routes are honored, SMTP will include its host name (for example, `VMHOST1`) in the path information, and will supply the following path for its MAIL FROM: command:

```
<@HOST1,@HOST2,@HOST3,@VMHOST1:USER@HOST4>
```

If such source routes are not honored, the list of hosts is removed from the mail routing path. In addition, SMTP will *not* add its host name to the path information.

```
     ┌─SOURCEROUTES YES───────────────────┐
►►───┼────────────────────────────────────┼──────────────────────────────►◄
     │                ┌─RCPTTO───┐        │
     └─SOURCEROUTES───┼──────────┼──NO────┘
                      ├─MAILFROM─┤
                      └─BOTH─────┘
```

Operands

YES
Indicates that source routes on the MAIL FROM: and RCPT TO: commands received from clients will be honored when mail is forwarded by SMTP. For the previous sample source route, SMTP will send the mail to HOST1 for further processing by HOST1. This is the default.

NO
Indicates that source routing is not to be honored. By default, only source routes for the RCPT TO: command will be ignored. The **RCPTTO**, **MAILFROM** or **BOTH** parameter can be specified with this parameter to select specific source routes to be ignored by SMTP.

Note: The **NO** parameter does *not* cause mail containing source routes to be rejected.

RCPTTO
Indicates that source routing is not to be honored for the RCPT TO: command. This is the default when NO is in effect. For the RCPT TO: command, any host list will be ignored and only the destination host will be used. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, SMTP will send the mail directly to USER@HOST4; the HOST1, HOST2 and HOST3 hosts will be ignored.

MAILFROM
Indicates that source routing is not to be honored for the MAIL FROM: command. When this parameter is used, the list of hosts will be removed from the mail routing path. In addition, SMTP will *not* add its host name to the path information. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, SMTP will supply the following MAIL FROM: command when mail is forwarded:

```
MAIL FROM: <USER@HOST4>
```

BOTH
Indicates that source routing is not to be honored for either of the RCPT TO: or the MAIL FROM: commands. Source routing information for these commands will be ignored as previously described.

SUPRESSNOTIFICATION Statement

The SUPRESSNOTIFICATION statement specifies that SMTP should not send a notification to the mail sender when mail delivery is successful. A notification will still be sent to the mail sender if mail delivery is unsuccessful.

```
►►──SUPRESSNOTIFICATION──────────────────────────────────────────────►◄
```

Operands

The SUPRESSNOTIFICATION statement has no operands.

TEMPERRORRETRIES Statement

The TEMPERRORRETRIES statement specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem. Temporary problems include network congestion, network connectivity, or a broken remote mail server.

```
    ┌─TEMPERRORRETRIES 0───────┐
►►──┴─TEMPERRORRETRIES retries─┴────────────────────────────────────►◄
```

Operands

retries
: The number of times mail delivery to a host with a temporary problem is retried. The default is 0, meaning that mail delivery is retried for the amount of time specified in the RETRYAGE statement. If *retries* is greater than 0, mail delivery is retried for that number of times. If delivery is still unsuccessful, the mail is returned to the sender. Change the default only when remote mail servers repeatedly terminate abnormally or hang SMTP mail transactions.

TRACE Statement

The TRACE statement specifies which type of tracing should be enabled during server initialization.

```
►►──TRACE──┬─ALL──────┬─────────────────────────────────────────────►◄
           ├─CODEFLOW─┤
           ├─CONN─────┤
           ├─DEBUG────┤
           ├─NOTICE───┤
           ├─RESOLVER─┤
           └─SPL──────┘
```

Operands

ALL
Initiates tracing of all types.

CODEFLOW
Initiates tracing of SMTP code flow.

CONN
Initiates tracing of connection activity.

DEBUG
Initiates tracing of all commands and replies, and their associated connection number (this same information was previously captured using the DEBUG configuration option provided with prior releases).

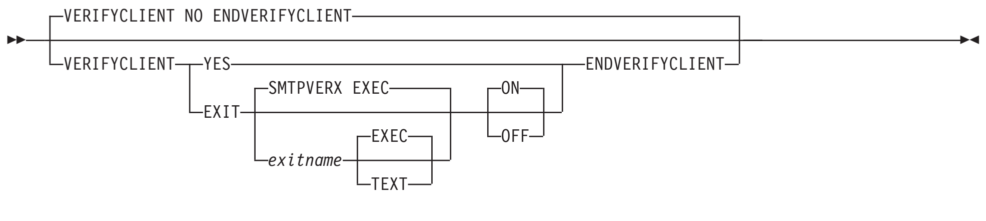
NOTICE
Initiates tracing of all TCP/IP notification events that are received by the SMTP virtual machine.

RESOLVER
Initiates resolver tracing. The same information can be produced by adding the **TRACE RESOLVER** statement to the TCPIP DATA file that is read by the SMTP server virtual machine at initialization.

SPL
Initiates tracing of *SPL IUCV execution.

VERIFYCLIENT Statement

The VERIFYCLIENT statement is used to indicate to the SMTP server whether or not client verification is to be performed. Client verification can be performed using the built-in client verification function (VERIFYCLIENT YES) or using a user exit (VERIFYCLIENT EXIT). For information on using the client verification exit, see the *TCP/IP Programmer's Reference*.



Operands

NO
Indicates that no client verification is to be performed; this is the default.

YES
Indicates verification of the client name specified on the HELO or EHLO command is to be performed using the built-in client verification function.

EXIT
Indicates a client verification exit routine is being defined.

exitname
Indicates the name of the exit routine associated with this command; the default exit name is **SMTPVERX**.

EXEC
Indicates the exit routine name specified on this command is the name of an EXEC; this is the default.

TEXT
Indicates the exit routine name specified on this command is the name of the text deck.

ON
Indicates the specified exit (being defined with this command) is to be enabled (on).

OFF
Indicates the specified exit (being defined with this command) is to be disabled (off).

Examples

- The following configuration file entry will enable the client verification exit routine SMTPVERX EXEC; this exit will perform all verification.

  ```
  VerifyClient
     exit smtpverx exec on
  EndVerifyClient
  ```

  When this entry is processed, the following text is displayed during server initialization:

  ```
  Client Verification            : Exit SMTPVERX EXEC ON
  ```

- This next entry defines the client verification exit routine SMTPVERX TEXT, but disables its use once SMTP server initialization is complete. Thus, client verification will not occur.

  ```
  VerifyClient
     exit smtpverx text off
  EndVerifyClient
  ```

  When this entry is processed, the following text is displayed during server initialization:

  ```
  Client Verification            : No (Exit SMTPVERX TEXT OFF)
  ```

VERIFYCLIENTDELAY Statement

The VERIFYCLIENTDELAY statement specifies the amount of time (in seconds) that SMTP will wait for the domain name system to respond during client verification. The default is 10 seconds.

If the client cannot be verified within the timeout, the mail item will be treated as if verification were not active. No comment is inserted into the mail header.

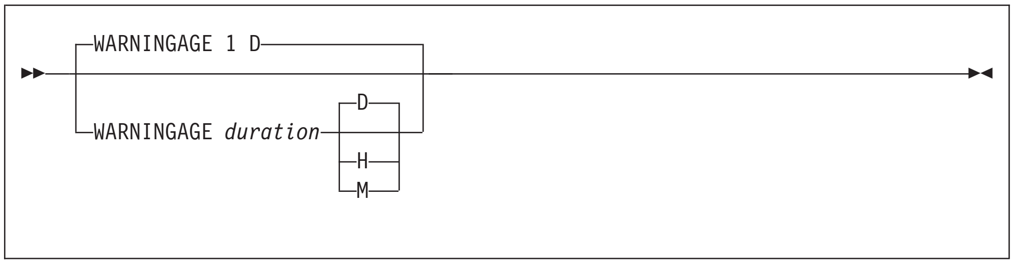
```
    ┌─VERIFYCLIENTDELAY 10──────┐
►►──┤                           ├────────────────────────────────────────►◄
    └─VERIFYCLIENTDELAY seconds─┘
```

Operands

seconds
: The number of seconds SMTP will wait for the domain name system to respond during client verification. The range is 0 through 86,400 seconds (24 hours). The default is 10 seconds.

WARNINGAGE Statement

The WARNINGAGE statement specifies the amount of time after which a copy of the mail is returned to the sender. The copy of the mail includes a header from SMTP that indicates the mail has been undeliverable for WARNINGAGE amount of time and that SMTP will continue to retry delivery of the mail for RETRYAGE amount of time.



Operands

duration
: The amount of time (specified in days, hours, or minutes) over which SMTP is to attempt delivery of a piece of mail before sending a non-delivery warning to the sender. If *duration* is specified in days, the range is 0 through 365; when it is given in hours, the range is 0 through 8760 (365 days). When *duration* is specified in minutes, the range is 0 through 525600 (365 days). The default is 1 day.

D Indicates that *duration* is specified in days. This is the default.

H Indicates that *duration* is specified in hours.

M Indicates that *duration* is specified in minutes.

When the RETRYAGE and the WARNINGAGE are equal, a warning is not issued to the sender. The warning is only issued if the WARNINGAGE is less than the RETRYAGE.

8BITMIME Statement

The 8BITMIME statement specifies the file name of the translation table to be used for 8-bit MIME support; the file type of this file must be TCPXLBIN. The translation table specified on this statement will be used *only* for 8-bit MIME support. Thus, if this statement is not specified, 8-bit MIME will not be supported.

```
►►──8BITMIME──filename──────────────────────────────────────────────►◄
```

Operands

filename
The file name of the translation table to be used for 8-bit MIME support.

See "Chapter 29. Using Translation Tables" on page 595 for more information.

Note: Changing the 8-bit MIME translation table could affect mail for which delivery is pending. When SMTP is restarted, notes that require 8-bit MIME support *and* are destined for non-RSCS network recipients will be processed using the new translation table.

Step 7: Advanced Configuration Considerations

Before you complete the SMTP servers configuration process, you may want to review the information in the following sections, in addition to the server exit information that follows:

- "Configuring a TCP/IP-to-RSCS Mail Gateway" on page 473
- "Configuring a TCP/IP-to-RSCS Secure Mail Gateway" on page 475
- "Defining Nicknames and Mailing Lists" on page 478
- "Customizing SMTP Mail Headers" on page 479

SMTP Server Exits

Several SMTP server exits are supported that allow for greater control over each piece of mail that is processed by the SMTP server. These exits are:

- the client verification exit, which might be used to reject mail from a particular host, designate certain trusted cites as "verified" but perform validation on all others, or control which users can use a particular SMTP server.
- the mail forwarding exit, which could be used to disallow forwarding of mail from a known sender of "junk" mail, intercept mail from specific clients and forward that mail to a local VM user ID for further analysis, or restrict the ability to forward mail to a particular set of hosts.
- the SMTP command exit, which allows control over specific SMTP commands. This exit might be used to reject particular SMTP commands, handle the delivery of local mail in a specific manner, or screen and reject mail based on content.

For more information on how to effectively use the exit routines mentioned above, see *z/VM: TCP/IP Level 3A0 Programmer's Reference, SC24-5983*.

Configuring a TCP/IP-to-RSCS Mail Gateway

You can configure the SMTP virtual machine with the GATEWAY option to run as a mail gateway between TCP/IP network users and users located on an RSCS network attached to the local host. Figure 18 on page 474 shows the SMTP virtual machine configured as a mail gateway.

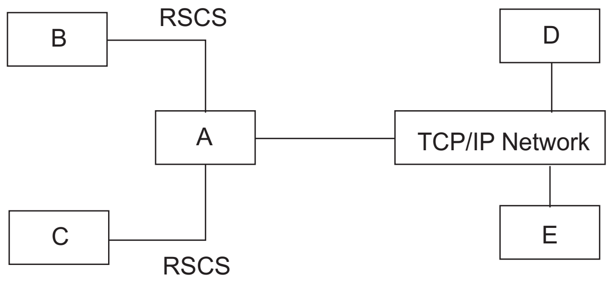


Figure 18. The SMTP Virtual Machine Configured as a Mail Gateway

Host	**Description**
A	The local VM host, running both TCP/IP and RSCS.
B and C	The hosts attached to host A through an RSCS network.
D and E	The hosts attached to host A through a TCP/IP network.

Users on hosts A, B, and C can send mail or files to users on TCP/IP hosts D and E using the CMS NOTE or SENDFILE commands, or using OfficeVision/VM™. For more information, see the *TCP/IP User's Guide*. The following steps describe how to configure a TCP/IP-to-RSCS mail gateway.

1. Update the SMTP configuration file to include the GATEWAY, RSCSDOMAIN, and RSCSFORMAT statements. Specify the GATEWAY, RSCSDOMAIN, and RSCSFORMAT options in the configuration file.
2. Enter the SMTPRSCS command, using the following format:

```
►►──SMTPRSCS──filename filetype filemode──────────────────────────────►◄
```

Operands

filename filetype filemode
The file identifier of the RSCS configuration file, for example, RSCS CONFIG A. The command requires a file name, file type and file mode.

This command creates the RSCS host table file, SMTPRSCS HOSTINFO. After the file is created, copy it to an SMTP server visible minidisk, such as TCPMAINT 198.

Note: If the local system's RSCS configuration file does not explicitly specify all of your RSCS network's nodes (for example, if it uses a default ROUTE * statement), you should either obtain the RSCS configuration file that does contain all of the RSCS network nodes and use that file as the SMTPRSCS command input, or create a local RSCS configuration file with ROUTE statements that do identify all of your RSCS network's nodes and use that as input to the SMTPRSCS command.

In order for an SMTP to RSCS gateway to work correctly, the SMTPRSCS HOSTINFO file must contain all RSCS nodes that SMTP mail could be destined for.

Perform the following on each RSCS node that sends mail through the SMTP virtual machine on a remote gateway node.

1. For each system running z/VM 3.1.0 or higher, place a TCPIP DATA file on each non-gateway VM system, on visible file space (for example the 190 disk). This DATA file must contain a SMTPSERVERID statement that identifies the user ID and RSCS node of your SMTP gateway. Copy the SMTPQUEU EXEC to each system on user visible file space as well.
2. For each non-gateway system running VM at a level prior to z/VM 3.1.0, if you have a previously defined SMTP mail gateway, and you installed the necessary files on the non-gateway VM systems, then no action is necessary on those systems.

   If you never installed the TCP/IP specific NOTE and SENDFILE functions on the non-gateway VM systems, then you need to do the following:

   a. Download the Mail Gateway package, MAILGATE VMARC. The URL www.ibm.com/s390/vm/related/tcpip (the VM TCP/IP home page) provides access to this package and instructions on how to extract files from the VMARC archive. Once the individual files are extracted, use the instructions to install and configure the gateway code (in the MAILGATE README file). The code you install is part of VM TCP/IP 2.4.0 (the NOTE and SENDFILE functions).
   b. Copy the SMTPQUEU EXEC to each VM system.
   c. Copy the gateway's SMTPRSCS HOSTINFO file to each system. See the SMTPRSCS command above (step 2 on page 474) for details on creating this file.

Configuring a TCP/IP-to-RSCS Secure Mail Gateway

The SMTP virtual machine can be configured with the SECURE statement to run as a secure mail gateway between TCP/IP network users and users located on an RSCS network attached to the local host. For information about how to set up a mail gateway, see "Configuring a TCP/IP-to-RSCS Mail Gateway" on page 473.

To enable the SMTP Secure Gateway mode, you must add the SECURE statement to the SMTP CONFIG file. When operating in Secure Gateway mode, only those RSCS addresses in the SMTP Security Table are authorized to send or receive mail. In addition, source routing is disabled to prevent the gateway from relaying mail to unauthorized users.

SMTP rejects mail to or from an unauthorized RSCS user. If the mail is from the TCP/IP network, SMTP rejects the RCPT TO: command with the error `550 User is not a registered gateway user`. If the mail is from the RSCS network, SMTP rejects the MAIL FROM: command with the error `550 User is not a registered gateway user`, and includes the file SECURITY MEMO as an explanation. For more information, see the examples of rejected mail and the sample SECURITY MEMO file on page 477.

Creating an SMTP Security Table

Create a file called SMTP SECTABLE that contains a list of RSCS users who are authorized to use the gateway. This file can have either fixed or variable length records of up to 255 characters in length. Records whose first non-blank character is an asterisk (*) are treated as comments and are ignored. The following example shows the format of the security table.

```
►►──RSCS_user_id──RSCS_node_id──────────────────────────────────────►

►──┬────────────────────────────────────────┬──────────────────────►◄
   └─nickname───primary_nick?───primary_mbox?─┘
```

Operands

RSCS_user_id
: The RSCS user ID of the authorized user.

RSCS_node_id
: The RSCS node ID of the authorized user.

nickname
: The name by which the RSCS user is known on the TCP/IP side of the gateway.

primary_nick?
: A primary nickname indicator, specified as **Y** or **N**. If Y, then mail addressed to *nickname*`@smtp-gateway` is automatically forwarded to *RSCS_user_id* `at` *RSCS_node_id*. Each nickname can only have one *primary_nick* record set to Y.

primary_mbox?
: A primary mailbox indicator, **Y** or **N**. If Y, then mail from *RSCS_user_id* `at` *RSCS_node_id* is converted to *nickname*`@smtp-gateway` before it is sent to the TCP/IP recipient. Each *RSCS_user_id*, *RSCS_node_id* pair can have only one *primary_mbox?* record.

A sample security table file is supplied on the TCPMAINT 591 disk as SMTPSECT SAMPTABL. Your customized table should stored on the TCPMAINT 198 disk as file SMTP SECTABLE.

The following is an example of an SMTPSECT SECTABL security table.

```
*  Records for Jane Doe, within IBM
JDOE   ALMADEN
JDOE   AUSTIN
*  Records for John Smith, within IBM
SMITH  RALEIGH   JOHNNY  Y  N
SMITH  YORKTOWN  JOHNNY  N  Y
SMITH  DALLAS    JOHNNY  N  N
SMITH  RALEIGH   JSMITH  Y  Y
```

For example, mail sent from the following RSCS network addresses through the SMTP gateway is rewritten to the following TCP/IP network addresses. Assume the host name of the gateway is `SMTP-GATEWAY.IBM.COM`.

```
RSCS Address         TCP/IP Address
-------------------------------------------------------------------------

JDOE  at ALMADEN     JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM
JDOE  at AUSTIN      JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM
SMITH at RALEIGH     JSMITH@SMTP-GATEWAY.IBM.COM
SMITH at YORKTOWN    JOHNNY@SMTP-GATEWAY.IBM.COM
SMITH at DALLAS      JOHNNY%DALLAS@SMTP-GATEWAY.IBM.COM
```

Mail sent from the TCP/IP network to the following TCP/IP network addresses, is forwarded to the following RSCS network addresses. Assume the host name of the gateway is smtp-gateway.ibm.com.

```
TCP/IP Address                             RSCS Address
------------------------------------------------------------------------

JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM     JDOE  at ALMADEN
JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM      JDOE  at AUSTIN
JSMITH@SMTP-GATEWAY.IBM.COM           SMITH at RALEIGH
JOHNNY@SMTP-GATEWAY.IBM.COM           SMITH at RALEIGH
MITH%DALLAS@SMTP-GATEWAY.IBM.COM      SMITH at DALLAS
```

| A sample security memo file is supplied on the TCPMAINT 591 disk as SMTPMEMO SAMPLE. Your customized memo should be stored on the TCPMAINT 198 disk as file SECURITY MEMO. The supplied sample memo file contains the following text:

```
The mail you sent to this SMTP gateway cannot be delivered because
you are not a registered user of this gateway.  Contact your local
administrator for instructions on how to be authorized to use this
SMTP gateway.
```

The following is an example of rejected mail that was sent to an unregistered RSCS user.

```
Date: Fri, 9 Oct 92 10:55:59 EST
From: SMTP@VM1.ACME.COM
To: DANIEL@VM1
Subject: Undeliverable Mail

VM1.ACME.COM unable to deliver following mail to recipient(s):
    <MATT@SMTP-GATEWAY.IBM.COM>
VM1.ACME.COM received negative reply from host:
    SMTP-GATEWAY
550 User 'MATT@SMTP-GATEWAY' is not a registered gateway user

          ** Text of Mail follows **
Date: Fri, 9 Oct 92 10:55:56 EDT
From: <DANIEL@VM1.ACME.COM>
To:   <MATT@SMTP-GATEWAY.IBM.COM>
Subject:  Lunch

Matt,

Do you have time to meet for lunch next week?  I want to discuss the
shipment of ACME iron birdseed.

Daniel
```

The following is an example of rejected mail that was sent from an unregistered RSCS user.

```
Date: Fri, 9 Oct 92 11:35:18 EST
From: <SMTP@SMTP-GATEWAY.IBM.COM>
To: <MATT@SMTP-GATEWAY.IBM.COM>
Subject: Undeliverable Mail

Unable to deliver mail to some/all recipients.
550 MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
550-User 'MATT@SMTP-GATEWAY' is not a registered gateway user.
550-
550-The mail you sent to this SMTP gateway cannot be delivered because
550-you are not a registered user of this gateway.  Contact your local
550-administrator for instructions on how to be authorized to use this
550-SMTP gateway.
```

```
           ** Text of Mail follows **
HELO SMTP-GATEWAY.IBM.COM
MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
RCPT TO:<DANIEL@VM1.ACME.COM>
DATA
Date: Fri, 9 Oct 92 11:34:17 EST
From: <MATT@SMTP-GATEWAY.IBM.COM>
To:   <DANIEL@VM1.ACME.COM>
Subject:  Awaiting your message

Daniel,

When are you going to contact me about the iron birdseed and giant
electromagnet that I ordered?  I would like to meet with you soon.

Matt

.
QUIT
```

Defining Nicknames and Mailing Lists

You can use the SMTP NAMES file to set up mail aliases, forwarding, and distribution lists. The following example illustrates the use of the SMTP NAMES file.

- Mail Aliases

  The entry:

  ```
  :nick.brat :userid.BART
  ```

  in the SMTP NAMES file on a TCP/IP host *ourvm.our.edu* causes all mail addressed to *brat@ourvm.our.edu* to be delivered to user ID BART on that host. *brat* becomes a mail alias for BART.
- Mail Forwarding

  The entry:

  ```
  :nick.homer :userid.HOMER :node.NEWVM
  ```

  in the SMTP NAMES file on TCP/IP host *ourvm.our.edu* (also known as OURVM on an RSCS network), causes all mail addressed to *homer@ourvm.our.edu* to be forwarded to HOMER at NEWVM on the RSCS network.
- Mail Distribution Lists

  The entry:

  ```
  :nick.princes
      :list.hal charles hamlet charming
              andrew at buc.acme.com  albert at win.ibm.com
  ```

  in the SMTP NAMES file on TCP/IP host *ourvm.our.edu* causes all mail addressed to *princes@ourvm.our.edu* to be sent to each recipient in the distribution list. Entries in the list can be addresses of recipients, or nicknames (defined in SMTP NAMES) of recipients.

Note: The nickname label may not be the same as any of the user IDs that are in the list defined by that nickname. Ensure that the nickname label is not used as one of the list entries.

Customizing SMTP Mail Headers

Electronic mail has a standardized syntax for text messages that are sent across networks. The standard syntax is described in RFC 822. Messages have an envelope and content. Fields in the envelope are in a rigid format and referred to as *headers*. Envelopes contain all necessary information to accomplish transmission and delivery of the message content.

The RFC 822 standard does not dictate the internal formats used at specific sites. IBM TCP/IP Level 3A0 allows specific sites to customize the SMTP mail headers with the REWRITE822HEADER statement.

You can use the REWRITE822HEADER statement to control the way SMTP performs a rewrite of the RFC 822 mail headers. Mail headers are passed from the local system or RSCS network to the TCP/IP network. Mail headers passing from the TCP/IP network to the local system or RSCS network are not affected. Mail envelope headers are also not affected. Only the addresses under certain RFC 822 header fields can be subject to the header rewriting rules.

The header fields affected by the REWRITE822HEADER statement are:

Fields	**Description**
From	The originator of the message.
Resent-From	The person that forwarded the message.
Reply-To	Provides a mechanism for indicating any mailboxes to which responses are to be sent.
Resent-Reply-To	The person to whom you should forward the reply.
Return-Path	Contains definitive information about the address and route back to the originator of the message. This field is added by the mail transport service at the time of final delivery.
Sender	The authenticated identity of the AGENT that sent the message. An AGENT can be a person, system, or process.
Resent-Sender	The authenticated identity of the AGENT that has resent the message.
To	Contains the identity of the primary recipient of the message.
Cc	Contains the identity of the secondary (informational) recipients of the message.
Bcc	Contains the identity of additional recipients of the message. The content of this field is not included in copies sent to the primary and secondary recipients of the message but included in the originator’s copy.

The SMTP RULES File

The SMTP RULES file contains the rewrite rules for the header addresses. You can create the SMTP RULES file during the configuration of the SMTP server to customize the address transformations to the needs of a particular site. Store the SMTP RULES file on TCPMAINT 198.

The SMTP RULES file consists of the following two sections:

FIELD DEFINITION
Contains the names of all header fields whose addresses are to be rewritten.

RULES DEFINITION
Contains the rewrite rules for the header fields.

In creating the SMTP RULES file, you must follow several syntactical conventions. The conventions are:

- The file statements are free-format. Tokens can be separated by an arbitrary number of spaces, and statements can span an arbitrary number of lines. However, you must terminate every statement with a semicolon (;).
- A character string appearing within single quotes ('...') is case-insensitive. For example, 'abc' represents 'abc', 'Abc', 'ABC', and so on. The use of character strings is illustrated in the following sections.
- A character string appearing within double-quotes ("...") 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 quotes or double-quotes.
- Double-hyphens ("--") are used to begin a comment. The comment extends to the end of the line.

The following sections describe the components of the SMTP RULES file.

Format of the Field Definition Section

The field definition section is the first section in any SMTP RULES file. It defines any applicable alias fields, and it is introduced by the following heading:

```
Field Definition Section
```

This section allows similar fields to be grouped under an alias or common name. This name, or alias, is used to represent the field list. You can define an arbitrary number of aliases representing a set of field lists.

An alias name can be any alphanumeric sequence of characters that is not a predefined keyword within the SMTP rules (see below). However, the alias name `DefaultFields` is treated specially by the SMTP configuration interpreter. If `DefaultFields` is defined, and if a rule is written that does not specify an associated field alias, the rules interpreter assumes that `DefaultFields` is the associated field alias.

The alias definition within this section is of the following form:

```
alias_name = alias_definition; optional comment
```

where *alias_name* is the name of the alias and *alias_definition* is an expression describing which fields are to be grouped under this alias. This expression can be as simple as a single field name. For example:

```
MyAlias = ’To’;
```

The aliases can be a list or set of field names. The field names `’To’ ’From’ ’Cc’ ’Bcc’`, in the following example, are part of a set of field names referenced by the alias `MyAlias`.

```
MyAlias = ’To’ ’From’ ’Cc’ ’Bcc’ ; -- first list of fields
```

You can combine field names and previously defined aliases to create a new alias. In the following example, the set of field names defined as `MyAlias` and the field names in the new alias `YourAlias` are combined to form a third set. The new alias `TheirAlias` is the union of both aliases and contain the fields of `MyAlias` and `YourAlias`.

```
MyAlias    = 'To' 'From' 'Cc' 'Bcc';
YourAlias  = 'Errors-To' 'Warnings-To';
TheirAlias =  MyAlias YourAlias;
```

In the previous example, `TheirAlias` is an alias that represents the following fields.

```
TheirAlias:  'To' 'From' 'Cc' 'Bcc' 'Errors-To' 'Warnings-To'
```

You can perform the following set algebra operations on set members of the alias to create a subset of the initial alias.

- Union operations
- Difference operations
- Intersection operations.

Union and Difference Operations: You can add or omit certain field names to a new alias of field names by using a minus sign to omit set members and an optional plus sign to include another field name. In the mathematics of sets, when you add together two or more sets, they form a union. When set members are omitted, the remaining set is created by the difference operation. In the following example `HerAlias` and `HisAlias` are defined. The alias `HisAlias` is created from the union of `TheirAlias`, `HerAlias`, and the omission of `Warning-To` and `Bcc` from the following sets.

```
HerAlias   = 'Reply-To' 'Sender';
HisAlias   = TheirAlias - 'Warnings-To' - 'Bcc' + HerAlias;
```

In the previous example, `HisAlias` is an alias that represents following fields.

```
HisAlias:  'To' 'From' 'Cc' 'Errors-To' 'Reply-To' 'Sender'
```

Intersection Operations: In addition to the union and difference operations previously shown, a field definition can include an intersection operation. When the intersection operation is applied to two field expressions, the resulting set contains the fields common to both. In the following example, `MyAlias` and `YourAlias` are defined. The alias `OurAlias` is created from the intersection of `MyAlias` and `YourAlias`. The asterisk (*) is the intersection operator.

```
MyAlias    = 'Bcc' 'Cc' 'From' 'Reply-To';
YourAlias  = 'Resent-From' 'Cc' 'Sender' 'To' 'Bcc';
OurAlias   = MyAlias * YourAlias; -- the intersection
```

In the previous example, `OurAlias` represents the following fields.

```
OurAlias:  'Bcc' 'Cc'
```

In the following complex example, `TheirAlias` is created from the intersection of `YourAlias` with the sum of `MyAlias` plus `Resent-From`.

```
TheirAlias = (MyAlias + 'Resent-From') * YourAlias;
```

In the previous example, `TheirAlias` represents the following fields.

```
TheirAlias: 'Bcc' 'Cc' 'Resent-From'
```

The parentheses within the definition of `TheirAlias` perform the same functions as in algebra. Field expressions are evaluated from left to right, but the intersection operation has greater priority than union and difference operations. If parentheses were not used in the definition of `TheirAlias`, the result would be:

```
TheirAlias: ’Bcc’ ’Cc’ ’From’ ’Reply-To’ ’Resent-From’
```

Format of the Rule Definition Section

The rule definition section is the next section in any SMTP RULES file. It contains the header rewriting rules that define the intended address transformations, and it is introduced by the following heading.

```
Rule Definition Section
```

The rewrite rules are given using a simple, free-format pattern matching language. The basic form of each rule is:

```
alias :before-address-pattern  => after-address-pattern;
```

The alias name *alias* is an optional name representing the fields for which the rule is applicable. If the alias name *alias*: is omitted from this part of the rules, then `DefaultFields` is assumed.

The sequence of tokens that define how a particular type of address is to be recognized is the *before-address-pattern* portion of the rules definition. The sequence of tokens that define how the address is to appear after the address has been rewritten is the *after-address-pattern* portion of the rules definition. The following example is the rule for converting host names.

```
A ’@’ RSCSHostName => A ’@’ TCPHostName; -- convert host names
```

In the previous example, `A ’@’ RSCSHostName` is the *before-address-pattern* portion of this rule, and `A ’@’ TCPHostName` is the *after-address-pattern* portion. This rule specifies that the address to be rewritten has an arbitrary local name (`A`), an at-sign, and the RSCS host name (`RSCSHostName`) of the current site. The rule also specifies that the rewritten address must contain the same arbitrary local name (`A`), an at-sign, and the current site’s TCP host name `TCPHostName`.

Syntax Convention of the SMTP Rules

The previous example of the rewriting rules shows that you must follow several syntactical conventions. The conventions are:

- Some keywords have special meaning to the rules interpreter. For example, `RSCSHostName` keyword means the RSCS host name of the present system, and `TCPHostName` keyword means the TCP host name of the present system. For more information about valid keywords, see “Predefined Keywords within the SMTP Rules” on page 484. Some keywords, such as `TCPHostName`, have single values. Other keywords, such as `AltTCPHostName` and `AnyDomainName`, can have many possible values. To avoid ambiguity, any keyword that can have multiple values, and is used in the *after-address-pattern* of a given rule, must appear exactly once within the *before-address-pattern* of that rule. The following rule example shows a valid syntax:

  ```
  A ’@’ AltTCPHostName ’.’ AltTCPHostName =>
     A ’%’ TCPHostName ’@’ TCPHostName;
  ```

  The following two rules have invalid syntax because the first keyword `AltTCPHostName` must be rewritten to a keyword with specific values. The `AltTCPHostName` is attempting to be rewritten to the same `AltTCPHostName` but with unknown values and becomes invalid.

```
A '@' AltTCPHostName '.' AltTCPHostName =>
   A '%' AltTCPHostName '@' TCPHostName;

         A '@' TCPHostName => A '@' AltTCPHostName;
```

Any rule whose *before-address-pattern* includes a keyword that has a null value is ignored during the header rewriting. Thus, if there is no `AltRSCSDomain` defined in the system configuration file, no rule that includes `AltRSCSDomain` in the *before-address-pattern* is considered during the header rewriting.

- Alphanumeric identifiers that are not within apostrophes or double-quotes, and that are not predefined keywords, are considered wildcards in the rule statement. Wildcards represent an arbitrary (non-null) sequence of characters. The identifier `A`, in the previous rule example, is a wildcard. Thus, if `host` were the RSCS host name for the current site, and if `tcphost` were the TCP host name for the current site, the previous rule example recognizes `abc@host` and `d@host` as candidates for address rewriting, and rewrites them as `abc@tcphost` and `d@tcphost` respectively. To avoid ambiguity, no two wildcards are allowed in a row, and the same wildcard cannot be used more than once, within the *before-address-pattern* of a given rule. The following rules have valid syntax:

  ```
  A '@' B TCPHostName      => A '%' B '@' TCPHostName;
  A '%' B '@' RSCSHostName => A B '@' TCPHostName;
  ```

  The following rules have invalid syntax because the first rule has two wildcards in a row `A` and `B`. The second rule has the same wildcard `A` repeated:

  ```
  A B '@' TCPHostName      => A A '%' B '@' TCPHostName;
  A '%' A '@' RSCSHostName => A '@' TCPHostName;
  ```

- A character string appearing within apostrophes or double-quotes tells the rules interpreter where a particular string is to appear within a header address. In the previous rule example, the '`@`' string in the *before-address-pattern* tells the rules interpreter that an at-sign must appear between the arbitrary character string and the RSCS host name. The '`@`' string in the *after-address-pattern* tells the rules interpreter that the address must be rewritten so an at-sign appears between the arbitrary string and the TCP host name. As previously mentioned, apostrophes denote case-insensitive strings, and double-quotes denote case-sensitive strings.
- The character sequence "`=>`", with no spaces between the characters separates the *before-address-pattern* from the *after-address-pattern*.
- The order in which the rules are specified is important; the first rule encountered whose *before-address-pattern* matches the current address is the rule to dictate the address transformation. Once a matching rule has been found for an address, no other rule is considered.

In addition to the rules themselves, there is the capability for some simple logic to decide at system configuration time which rules within the file should become active. These conditions are specified in the form of an IF-THEN-ELSE statement as shown in the following example.

```
IF cond THEN
   statement list
ELSE
   statement list
ENDIF
```

A statement list can consist of any number of rules or nested IF statements, or both. Each IF statement, regardless of whether it is nested, must be terminated by an ENDIF keyword. As with IF statements in other languages, the ELSE clause is optional. There are only two conditions recognized by an IF statement:

- IF *predefined keyword* = '*character string*' THEN

- IF *predefined keyword* CONTAINS '*character string*' THEN ... ENDIF

The conditional operators = and CONTAINS can be prefixed by the word NOT to invert the conditions.

The *predefined keyword* must be a keyword that resolves to a single value at system configuration time. The character string in the first condition can be null. A character string cannot span more than one line.

The following is an example of the use of IF statements.

```
IF RSCSDomain = '' THEN
   A '@' AnyRSCSHostName => A '%' AnyRSCSHostName '@' TCPHostName;
ELSE
   A '@' RSCSHostName '.' RSCSDomain    => A '@' TCPHostName;
   A '@' RSCSHostName '.' AltRSCSDomain => A '@' TCPHostName;
   IF RSCSDomain CONTAINS '.' THEN
      A '@' AnyRSCSHostName                  =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
      A '@' AnyRSCSHostName '.' RSCSDomain    =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
      A '@' AnyRSCSHostName '.' AltRSCSDomain =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
   ELSE
      A '@' AnyRSCSHostName                  =>
         A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
      A '@' AnyRSCSHostName '.' RSCSDomain    =>
         A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
      A '@' AnyRSCSHostName '.' AltRSCSDomain =>
         A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
   ENDIF
ENDIF
```

Predefined Keywords within the SMTP Rules

You can use the following predefined keywords to define the header rewriting rules.

Keyword	**Definition**
TCPHostName	Matches the TCP host name of the system as defined by the concatenation of the HOSTNAME and DOMAINORIGIN statements in the TCPIP DATA file.
TCPHostNameDomain	Matches the domain portion of the TCP host name of the system as defined by the DOMAINORIGIN statement in the TCPIP DATA file. For example, if the TCP host name was `vm1.acme.com`, the value of TCPHostNameDomain is `acme.com`.
ShortTCPHostName	Matches the first portion of the TCP host name of the system, as defined by the HOSTNAME statement in the TCPIP DATA file. For example, if the TCP host name was `vm1.acme.com`, the value of ShortTCPHostName is `vm1`.
AltTCPHostName	Matches any alternative TCP host name of the system, as defined by ALTTCPHOSTNAME statements in the SMTP CONFIG file
RSCSHostName	Matches the RSCS host name of the system from the CMS IDENTIFY command. NJEHostName is a synonym for RSCSHostName.

AnyRSCSHostName	Matches any (unqualified) RSCS host name defined in the SMTPRSCS HOSTINFO file. AnyNJEHostName is a synonym for AnyRSCSHostName.
RSCSDomain	Matches the domain name of the RSCS network as defined by the RSCSDOMAIN statement in the SMTP CONFIG file. NJEDomain is a synonym for RSCSDomain.
AltRSCSDomain	Matches the alternative domain name of the RSCS network as defined by the ALTRSCSDOMAIN statement in the SMTP CONFIG file. AltNJEDomain is a synonym for AltRSCSDomain.
AnyDomainName	Matches any fully qualified domain name. Any host name with a period (.) is considered to be a fully qualified domain name.
SecureNickAddr	Matches an address of the form *RSCS_user_id@RSCS_node_id*, where *RSCS_user_id*, and *RSCS_node_id* are defined with a nickname in the SMTP SECTABLE file. **Note:** This matches only user and node IDs that are defined with nicknames. When SecureNickAddr is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickName with the corresponding nickname. This allows SecureNickName to be specified in the *after-address-pattern*.
SecureNickName	Matches a nickname defined in the SMTP SECTABLE file. When SecureNickName is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickAddr with the corresponding *RSCS_user_id@RSCS_node_id*. This allows SecureNickAddr to be specified in the *after-address-pattern*.

The predefined keywords defined previously can consist of any combination of uppercase and lowercase characters; the rules interpreter does not distinguish between them.

The secure keywords are only valid when SMTP is configured to be a secure gateway.

Default SMTP Rules

If the SMTP RULES file does not exist, SMTP uses a default set of rules. The default set used depends on whether SMTP is configured as a secure gateway.

SMTP Non-Secure Gateway Configuration Defaults

If SMTP is not configured as a secure gateway, SMTP uses the following default:

```
Field Definition Section

DefaultFields = 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
                'Resent-Reply-To' 'Resent-Sender' 'Return-Path'
                'Sender' 'To';

Rule Definition Section


A '@' RSCSHostName => A '@' TCPHostName;

IF RSCSDomain = '' THEN
   A '@' AnyRSCSHostName => A '%' AnyRSCSHostName '@' TCPHostName;
ELSE
   A '@' RSCSHostName '.' RSCSDomain    => A '@' TCPHostName;
   A '@' RSCSHostName '.' AltRSCSDomain => A '@' TCPHostName;
   IF RSCSDomain CONTAINS '.' THEN
      A '@' AnyRSCSHostName                   =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
      A '@' AnyRSCSHostName '.' RSCSDomain    =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
      A '@' AnyRSCSHostName '.' AltRSCSDomain =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
   ELSE
      A '@' AnyRSCSHostName                   =>
         A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
      A '@' AnyRSCSHostName '.' RSCSDomain    =>
         A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
      A '@' AnyRSCSHostName '.' AltRSCSDomain =>
         A '%' AnyRSCSHostName '.' RSCSDomain '@' TCPHostName;
   ENDIF
ENDIF

A '@' TCPHostName      => A '@' TCPHostName;
A '@' ShortTCPHostName => A '@' TCPHostName;
A '@' AltTCPHostName   => A '@' TCPHostName;
A '@' AnyDomainName    => A '@' AnyDomainName;
A '@' B                => A '@' B '.' TCPHostNameDomain;
```

SMTP Secure Gateway Configuration Defaults

If SMTP is configured as a secure gateway, SMTP uses the following default:

```
Field Definition Section

DefaultFields = 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
                'Resent-Reply-To' 'Resent-Sender' 'Return-Path'
                'Sender' 'To';

Rule Definition Section

SecureNickAddr     => SecureNickName '@' TCPHostName;
A '@' RSCSHostName => A '@' TCPHostName;

IF RSCSDomain NOT = '' THEN
   SecureNickAddr '.' RSCSDomain    => SecureNickName '@' TCPHostName;
   SecureNickAddr '.' AltRSCSDomain => SecureNickName '@' TCPHostName;
   A '@' RSCSHostName '.' RSCSDomain    => A '@' TCPHostName;
   A '@' RSCSHostName '.' AltRSCSDomain => A '@' TCPHostName;
   IF RSCSDomain CONTAINS '.' THEN
      A '@' AnyRSCSHostName                   =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
      A '@' AnyRSCSHostName '.' RSCSDomain    =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
      A '@' AnyRSCSHostName '.' AltRSCSDomain =>
         A '@' AnyRSCSHostName '.' RSCSDomain;
   ELSE
      A '@' AnyRSCSHostName                   =>
```

```
         A ’%’ AnyRSCSHostName ’.’ RSCSDomain ’@’ TCPHostName;
      A ’@’ AnyRSCSHostName ’.’ RSCSDomain    =>
         A ’%’ AnyRSCSHostName ’.’ RSCSDomain ’@’ TCPHostName;
      A ’@’ AnyRSCSHostName ’.’ AltRSCSDomain =>
         A ’%’ AnyRSCSHostName ’.’ RSCSDomain ’@’ TCPHostName;
   ENDIF
ENDIF

A ’@’ TCPHostName      => A ’@’ TCPHostName;
A ’@’ ShortTCPHostName => A ’@’ TCPHostName;
A ’@’ AltTCPHostName   => A ’@’ TCPHostName;
A ’@’ AnyDomainName    => A ’@’ AnyDomainName;
A ’@’ B                => A ’@’ B ’.’ TCPHostNameDomain;
```

Examples of Header Rewrite Rules

The following are examples of how the default header rewriting rules affect an SMTP mail header. The example site is not a secure gateway and is configured as shown in the following example.

```
TCPHostName      = vm1.acme.com
ShortTCPHostName = vm1
AltTCPHostName   = seeds.acme.com
RSCSHostName     = vm1
RSCSDomain       = acmenet
AltRSCSDomain    = centralnet
```

In addition, assume that the following are known to be other RSCS hosts:

```
bird
iron
```

Then the following header:

```
From: abc@vm1 (Brendan Beeper)
To: "Jenny Bird" <def@bird>
Cc: ghi@iron.acmenet, j@vm1,
 k@seeds.acme.com
Subject: New Ore
Sender: "Mailing List" <owner@acmenet>
Bcc: lmno@iron.centralnet
```

is rewritten as:

```
From: abc@vm1.acme.com (Brendan Beeper)
To: "Jenny Bird" <def%bird.acmenet@vm1.acme.com>
Cc: ghi%iron.acmenet@vm1.acme.com, j@vm1.acme.com,
 k@vm1.acme.com
Subject: New Ore
Sender: "Mailing List" <owner%acmenet@vm1.acme.com>
Bcc: lmno%iron.acmenet@vm1.acme.com
```

If you change the rule before the two ENDIFs to:

```
A ’@’ AnyRSCSHostName ’.’ AltRSCSDomain =>
   ’<@’ TCPHostName ’:’ A ’@’ AnyRSCSHostName ’.’ RSCSDomain ’>’;
```

then the original Bcc field within our header is rewritten as:

```
Bcc: <@vm1.acme.com:lmno@iron.acmenet>
```

Note: Do not make the change shown in the previous example; it is intended only as a demonstration of the capabilities of the pattern-matching language.

Dynamic Server Operation

The VM Special Message (SMSG) command provides an interactive interface to the SMTP server to:

- perform general user tasks such as querying the SMTP mail delivery queues and operating statistics of the SMTP server.
- perform privileged system administration tasks, such as rebooting or shutting down the SMTP server and enabling or disabling various tracing and debugging options.

Notes:

1. Privileged SMSG commands are accepted only from users that have been included on the SMTP server SMSGAUTHLIST configuration statement.
2. Command responses are returned to the originator of an SMSG command through the use of CP MSG commands (or CP MSGNOH commands if the SMTP server is running with CP privilege class B).

SMSG Interface to the SMTP Server

The various SMSG commands and operands supported by the name server are presented throughout the remainder of this section, with general-user commands presented separately from those that are available to only privileged users.

General User SMSG Commands

Use the SMSG command for general user queries for the mail delivery queues and the operating statistics of the SMTP server.

```
►►──SMSG──server_id──┬─HElp───┬──────────────────────────────────────►◄
                     ├─QUeues─┤
                     └─STats──┘
```

Operands

server_id
The user ID of the SMTP server virtual machine.

HElp
Provides a list of valid SMSG commands accepted by SMTP.

QUeues
Provides a list of mail queued on the various SMTP mail delivery queues.

STats
Provides operating statistics about the SMTP server.

Privileged User SMSG Commands

The privileged SMTP SMSG commands allow system administrators to perform SMTP administration tasks, such as rebooting or shutting down the SMTP server, controlling tracing activity, or altering specific aspects of SMTP server operation.

Only the most basic privileged SMSG commands are discussed here; the remaining SMSG commands are discussed in the sections listed in table Table 26.

```
►►──SMSG──server_id──┬─HElp─────┬──────────────────────────────────────────►◄
                     ├─CLosecon─┤
                     ├─REboot───┤
                     └─SHutdown─┘
```

Privileged user SMSG commands are accepted only from users specified with the SMSGAUTHLIST configuration statement.

Operands

server_id
The user ID of the SMTP server virtual machine.

HElp
Provides brief help about SMSG commands supported by the SMTP server.

CLosecon
Closes the SMTP server's console log and sends it to the `:Owner.` identified in the DTCPARMS file.

REboot
Causes SMTP to Initial Program Load (IPL) CMS.

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

Table 26 summarizes the additional privileged user SMTP SMSG commands.

Table 26. Privileged SMTP SMSG Commands

Command	Description	Page
SMSG FORWARDMAIL	Enables or disables mail forwarding, or identifies a user exit to be used to control mail forwarding.	490
SMSG LISTMAIL	Displays the number of pieces of mail or a list of mail currently being processed by the SMTP server.	491
SMSG MAILINFO	Displays general envelope information for a given piece of mail.	493
SMSG PURGE	Purges a single piece of mail.	495
SMSG REPROCESS	Causes a piece of mail to be reprocessed.	496
SMSG SMTPCMDS	Defines the characteristics of the SMTP command user exit.	496
SMSG SOURCEROUTES	Specifies whether or not the SMTP server is to honor source routes.	499
SMSG TRACE	Controls the tracing activity performed by the SMTP server.	501
SMSG VERIFYCLIENT	Specifies whether or not client verification is to be performed.	502

Privileged User SMSG FORWARDMAIL Command

The FORWARDMAIL command is used to enable or disable mail forwarding, to identify a user exit to be used to control such activity, and to query the current FORWARDMAIL setting. For information on using the mail forwarding exit, see the *TCP/IP Programmer's Reference*.

```
►►──SMSG ──server_id──FORWARDmail──┬─YES──────────────────────────────────────┬──────────►◄
                                   ├─NO───────────────────────────────────────┤
                                   │       ┌─SMTPFWDX EXEC────────┐   ┌─ON──┐ │
                                   ├─EXIT──┼──────────────────────┼───┼─────┼─┤
                                   │       │           ┌─EXEC──┐  │   └─OFF─┘ │
                                   │       └─exitname──┼───────┼──┘           │
                                   │                   └─TEXT──┘              │
                                   ├─EXIT──RELOAD─────────────────────────────┤
                                   └─QUERY────────────────────────────────────┘
```

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
: The user ID of the SMTP server virtual machine.

YES
: Indicates mail forwarding is to be performed.

NO
: Indicates that no mail forwarding is to be performed. When the SMTP server determines the recipient is not on the local system, the RCPT TO: command will be rejected.

EXIT
: Indicates a mail forwarding exit routine is being turned on or off by this command.

exitname
: The name of the exit routine associated with this command (the default exit routine name for FORWARDMAIL is SMTPFWDX).

EXEC
: Indicates the exit routine name specified on this command is the name of an EXEC (this is the default).

TEXT
: Indicates the exit routine name specified on this command is the name of a text deck.

ON
: Indicates the specified exit is being enabled (turned on).

OFF
: Indicates the specified exit is being disabled (turned off).

RELOAD
: Forces the exit routine to be reloaded the next time it is executed. If the exit routine is a REXX exec, it will be EXECLOADed into storage the next time it is executed.

QUERY
Returns current FORWARDMAIL settings; the returned response indicates whether mail forwarding is enabled or disabled. If an exit has been defined, the name of the exit is included in the response. Sample responses for several settings are shown in Table 27.

Table 27. Mail Forwarding Exit - Sample Queries

Mail Forwarding Setting	Response from Query
Mail Forwarding ON	`* From SMTP: FORWARDMAIL is set to YES`
Mail Forwarding OFF	`* From SMTP: FORWARDMAIL is set to NO`
Exit SMTPFWDX TEXT enabled	`* From SMTP: FORWARDMAIL exit SMTPFWDX TEXT ON`
Exit SMTPFWDX TEXT disabled	`* From SMTP: FORWARDMAIL is set to YES (Exit SMTPFWDX TEXT OFF)`

Examples

- The command that follows will enable the mail forwarding exit routine SMTPFWDX EXEC.

  ```
  smsg smtp forwardmail exit smtpfwdx exec on
  ```

  The following response is displayed:

  ```
  * From SMTP: FORWARDMAIL exit SMTPFWDX EXEC ON
  ```
- This next command will enable mail forwarding, and at the same time disable any mail forwarding exit that is currently in use.

  ```
  smsg smtp forwardmail yes
  ```

  The following response is displayed:

  ```
  * From SMTP: FORWARDMAIL is set to YES
  ```
- This last command will query the current mail forwarding setting.

  ```
  smsg smtp forwardmail query
  ```

  The following response is displayed:

  ```
  * From SMTP: FORWARDMAIL is set to YES (Exit SMTPFWDX EXEC OFF)
  ```

Privileged User SMSG LISTMAIL Command

The LISTMAIL command provides the number of pieces of mail and/or a list of mail currently being processed by the SMTP server. The list will be sorted from the oldest to the newest piece of mail and will contain the following information for each piece of mail: mail ID, sender ID, Total Recipients, and Mail State.

```
                              ┌─COUNT──┐
►►──SMSG──server_id──LISTMail─┼────────┼──────────────────────────────────►◄
                              ├─number─┤
                              └─ALL────┘
```

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
: The user ID of the SMTP server virtual machine.

COUNT
: Displays a count of the number of pieces of mail currently being processed by the SMTP server. COUNT is the default.

number
: The number of pieces of mail to display in the list, starting with the oldest piece of mail.

ALL
: Indicates that all mail will be displayed in the list, starting with the oldest piece of mail.

Mail States

Mail can be in one or more of the following mail states:

Receiving
: Mail envelope and text are currently being received by the SMTP server.

Resolving
: Mail name server resolution is currently being processed.

Spooling
: Mail is currently queued up for local delivery.

Waiting to Send
: Mail is currently queued up for delivery to a remote host. There is currently no open connection.

Sending
: Mail is currently queued up on an open connection to be sent to a remote host. The mail at the head of the queue is actively communicating with the remote host.

Waiting to Retry
: Mail is currently queued up to be delivered to a remote host again because an earlier delivery attempt was unsuccessful.

Holding
: Mail is currently waiting for operator intervention because local delivery was unsuccessful.

Note: Mail will remain in the *Holding* state until a REPROCESS, PURGE, REBOOT, or SHUTDOWN command is issued.

Examples

```
sm smtp listm
Ready; T=0.01/0.01 15:42:05
* From SMTP3: Total pieces of mail in process: 5

sm smtp listm count
Ready; T=0.01/0.01 15:42:50
* From SMTP3: Total pieces of mail in process: 5

sm smtp listm 3
Ready; T=0.01/0.01 15:43:22
* From SMTP3: Total pieces of mail in process: 5
* From SMTP3: Number requested: 3
* From SMTP3:
* From SMTP3:                                       Total  Mail
* From SMTP3: mail Id   Sender (first 25 chars)     Rcpts  State
* From SMTP3: --------  ------------------------  -------  ------
* From SMTP3: 1         USER1@VM.WATSON.IBM.COM         8  Waiting to Retry
* From SMTP3: 2         USER3@VM.WATSON.IBM.COM        12  Waiting to Send
* From SMTP3: 3         USER@VM.WATSON.IBM.COM          3  Sending

sm smtp listm all
Ready; T=0.01/0.01 15:44:01
* From SMTP3: Total pieces of mail in process: 5
* From SMTP3: Number requested: 5
* From SMTP3:
* From SMTP3:                                      Total  Mail
* From SMTP3: mail Id   Sender (first 25 chars)    Rcpts  State
* From SMTP3: --------  ------------------------  -------  ------
* From SMTP3: 1         USER1@VM.WATSON.IBM.COM         8  Waiting to Retry
* From SMTP3: 2         USER3@VM.WATSON.IBM.COM        12  Waiting to Send
* From SMTP3: 3         USER@VM.WATSON.IBM.COM          3  Sending
* From SMTP3: 5         jdoe@vm1.acme.com               1  Receiving+Resolving
* From SMTP3: 7         mikew@vm3.acme.com             14  Receiving
```

Privileged User SMSG MAILINFO Command

The MAILINFO command provides general envelope information for a given piece of mail.

```
►►──SMSG──server_id──MAILInfo──mailid──────────────────────────────────────►◄
```

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
The user ID of the SMTP server virtual machine.

mailid
The mail ID of the piece of mail for which mail information is to be displayed. The mail ID for a piece of mail may be obtained by using the SMSG LISTMAIL command.

General Envelope Information

The MAILINFO command provides the following general envelope information for a given piece of mail:

Total Rcpts
The total number of recipients for the piece of mail.

Unresolved Rcpts
The total number of unresolved recipients for the piece of mail.

Remaining Rcpts
The total number of remaining recipients for the piece of mail.

Date Received
The date the mail was received by the SMTP server.

Time Received
The time the mail was received by the SMTP server.

Mail Size
The size of the mail in bytes.

Sender ID
The user ID of the original sender of the mail.

The following four Batch File fields are provided for a given piece of mail only if the mail was generated as the result of another virtual machine having sent a Batch SMTP (BSMTP) file to the SMTP server virtual machine:

Note:

Source User ID
The user ID of the virtual machine from which the Batch SMTP file was spooled.

Source Node ID
The node ID of the virtual machine from which the Batch SMTP file was spooled.

Source Spool ID
The spool ID of the Batch SMTP file on the virtual machine that sent the file to SMTP.

Current Spool ID
The spool ID of the Batch SMTP file on the SMTP server virtual machine.

The following general envelope information may or may not be provided, as this information is dependent upon the status of the mail:

Sender Domain
The host domain of the original sender of the mail.

Mail From String
The sender path address specified with the SMTP MAIL FROM: command for this piece of mail.

Rcpt ID
The user ID of the recipient of the mail.

Rcpt Domain
The host domain of the recipient of the mail.

Rcpt To String
The recipient path address specified with the SMTP RCPT TO: command for this piece of mail.

IP Address
The resolved recipient IP address.

Rcpt Domain, *Rcpt ID*, *Rcpt To String*, and *IP address* will be given for each recipient of the mail.

Examples

```
sm smtp mailinf 3
Ready; T=0.01/0.01 15:44:01
* From SMTP3: Mail Information
* From SMTP3: ----------------
* From SMTP3: Total Rcpts       :  3
* From SMTP3: Unresolved Rcpts  :  0
* From SMTP3: Remaining Rcpts   :  3
* From SMTP3: Date Received     :  01/01/99
* From SMTP3: Time Received     :  15:41:52
* From SMTP3: Mail Size (bytes) :  212
* From SMTP3: Sender ID         :  USER
* From SMTP3: Sender Domain     :  VM
* From SMTP3: Mail From String  :  <USER@VM.WATSON.IBM.COM>
* From SMTP3:
* From SMTP3: Batch File
* From SMTP3:   Source User ID   :  USER
* From SMTP3:   Source Node ID   :  VM
* From SMTP3:   Source Spool ID  :  1008
* From SMTP3:   Current Spool ID :  0004
* From SMTP3:
* From SMTP3: Remaining Recipients
* From SMTP3:   Rcpt ID          :  user5
* From SMTP3:   Rcpt Domain      :  vm1.acme.com
* From SMTP3:   Rcpt To String   :  <user5@vm1.acme.com>
* From SMTP3:   IP Addresses     :  1.234.56.78
* From SMTP3:
* From SMTP3:   Rcpt ID          :  user4
* From SMTP3:   Rcpt Domain      :  vm1.acme.com
* From SMTP3:   Rcpt To String   :  <user4@vm1.acme.com>
* From SMTP3:   IP Addresses     :  12.567.8.90
* From SMTP3:                    :  9.876.54.32
* From SMTP3:
* From SMTP3:   Rcpt ID          :  teri
* From SMTP3:   Rcpt Domain      :  vm1.acme.com
* From SMTP3:   Rcpt To String   :  <teri@vm1.acme.com>
* From SMTP3:   IP Addresses     :  123.456.78.9
```

Privileged User SMSG PURGE Command

The PURGE command allows an authorized user to purge error mail, provided it is in one of these mail states: *Waiting to Send*, *Waiting to Retry*, *Spooling*, *Sending*, and/or *Holding*.

The original sender of the purged mail will be notified of this action through error mail.

Note: Mail that is actively communicating with a remote host cannot be purged; in this case, the mail must finish communicating with the remote host before the PURGE command may be issued.

```
►►──SMSG──server_id──PUrge──mailid──────────────────────────────────────►◄
```

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
The user ID of the SMTP server virtual machine.

mailid
The mail ID of the piece of mail to be purged. The Mail ID of a piece of mail may be obtained by using the LISTMAIL command.

Examples

```
sm smtp pu 1
Ready; T=0.01/0.01 15:46:38
```

Privileged User SMSG REPROCESS Command

The REPROCESS command allows an authorized user to force a piece of mail to be reprocessed, provided it is in one of these mail states: *Waiting to Send*, *Waiting to Retry*, *Spooling*, *Sending*, and/or *Holding*.

Note: Mail that is actively communicating with a remote host cannot be reprocessed; in this case, the mail must finish communicating with the remote host before the REPROCESS command may be issued.

The REPROCESS command may be useful when a particular piece of mail cannot be delivered for some period of time, during which the IP addresses in the mail have become old or obsolete. With REPROCESS, the recipient addresses for the mail will be newly resolved and delivery of the mail will again be attempted.

```
►►──SMSG──server_id──REProcess──mailid──────────────────────────────────────►◄
```

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
The user ID of the SMTP server virtual machine.

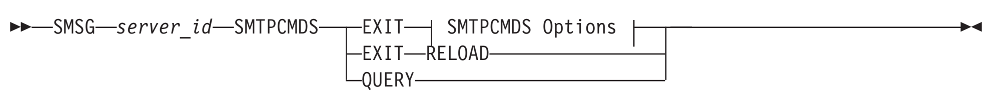
mailid
The mail ID of the piece of mail to be processed again. The Mail ID of a piece of mail may be obtained by using the LISTMAIL command.

Examples

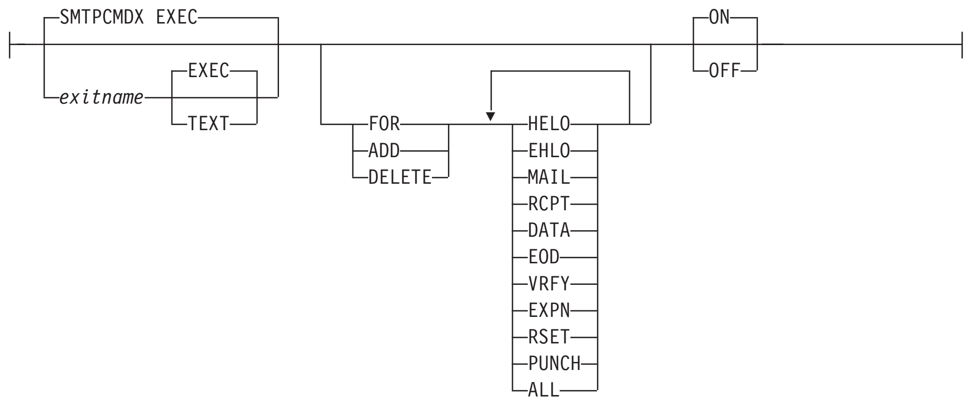
```
sm smtp reprocess 1
Ready; T=0.01/0.01 15:45:22
* From SMTP3: Mail Id 1 is being reprocessed.
```

Privileged User SMSG SMTPCMDS Command

The SMTPCMDS command is used to define the characteristics of the SMTP command user exit, to query those characteristics, and to indicate whether or not the exit is to be called by setting it on or off. For information on using the SMTP command exit, see the *TCP/IP Programmer's Reference*.



SMTPCMDS Options:



Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
: The user ID of the SMTP server virtual machine.

exitname
: The name of the exit routine associated with this command (the default exit routine name for SMTPCMDS is SMTPCMDX).

EXEC
: Indicates the exit routine name specified on this command is the name of an EXEC (this is the default).

TEXT
: Indicates the exit routine name specified on this command is the name of a text deck.

FOR
: Precedes the exact list of commands for which the exit is being defined.

ADD
: Precedes the list of commands that are to be added to this exit definition.

DELETE
: Precedes the list of commands that are to be deleted from this exit definition.

HELO
: Indicates the exit routine is to be turned on or off for the HELO command.

EHLO
: Indicates the exit routine is to be turned on or off for the EHLO command.

MAIL
: Indicates the exit routine is to be turned on or off for the MAIL FROM: command.

RCPT
Indicates the exit routine is to be turned on or off for the RCPT TO: command.

DATA
Indicates the exit routine is to be turned on or off for the DATA command.

EOD
Indicates the exit routine is to be turned on or off when the “end of data” condition is reached; that is, when a period (.) is received by the SMTP server after a DATA command.

VRFY
Indicates the exit routine is to be turned on or off for the VRFY command.

EXPN
Indicates the exit routine is to be turned on or off for the EXPN command.

RSET
Indicates the exit routine is to be turned on or off for the RSET command.

PUNCH
Indicates the exit routine is to be turned on or off for PUNCH processing. If the exit is enabled for this condition, it will be called when the SMTP server is ready to punch local mail (mail on the same node or RSCS network) to its destination.

ALL
Indicates the exit routine is to be turned on or off for all of the SMTP commands for which user exit capability is provided.

ON
Indicates the specified exit is being enabled (turned on) for a particular command or set of commands.

OFF
Indicates the specified exit is being disabled (turned off) for a particular command or set of commands.

RELOAD
Forces the exit routine to be reloaded the next time it is executed. If the exit routine is a REXX exec, it will be EXECLOADed into storage the next time it is executed.

QUERY
Returns current SMTPCMDS exit settings. The returned response indicates whether the command exit is enabled or disabled. If an exit has been defined, the name of the exit is included in the response. The commands associated with the current exit state are indicated as well. Sample responses for several settings are shown in Table 28.

Table 28. SMTP Command Exit - Sample Queries

SMTP Command Exit Setting	**Response from Query**
Exit not defined	`* From SMTP: SMTPCMDS exit not defined`
Exit SMTPCMDX EXEC enabled for HELO, VRFY, and EXPN	`* From SMTP: SMTPCMDS exit SMTPCMDX EXEC ON * From SMTP: SMTPCMDX defined for HELO VRFY EXPN`
Exit SMTPCMDX EXEC disabled	`* From SMTP: SMTPCMDS exit SMTPCMDX EXEC OFF * From SMTP: SMTPCMDX defined for HELO VRFY EXPN`

Examples

- The command that follows will enable the SMTP command exit routine SMTPCMDX EXEC for the SMTP HELO command:

  ```
  smsg smtp smtpcmds exit for helo
  ```

  The following response is displayed:

  ```
  * From SMTP: SMTPCMDS exit SMTPCMDX EXEC ON
  * From SMTP: SMTPCMDX defined for HELO
  ```
- The next command will add the SMTP commands VRFY and EXPN to the previous exit definition:

  ```
  smsg smtp smtpcmds exit add vrfy expn
  ```

  The following response is displayed:

  ```
  * From SMTP: SMTPCMDS exit SMTPCMDX EXEC ON
  * From SMTP: SMTPCMDX defined for HELO VRFY EXPN
  ```
- Lastly, the command that follows will disable the SMTP commands exit:

  ```
  smsg smtp smtpcmds exit off
  ```

  The following response is displayed:

  ```
   * From SMTP: SMTPCMDS exit SMTPCMDX EXEC OFF
   * From SMTP: SMTPCMDX defined for HELO VRFY EXPN
  ```

Privileged User SMSG SOURCEROUTES Command

The SOURCEROUTES command is used to specify whether the SMTP server will honor source routes, and to query the current SOURCEROUTES setting.

A source route is a path that contains a routing list of hosts and a destination mailbox. The list of hosts is the *route* — information about how the mail is to arrive at its final destination; the mail is passed from one host in this list to the next until it is delivered to the intended recipient.

The specification that follows is an example of a **source route**:

```
<@HOST1,@HOST2,@HOST3:USER@HOST4>
```

The list of hosts is HOST1, HOST2 and HOST3, and the destination is USER@HOST4.

If this sample source route is included with a RCPT TO: command, and is honored by SMTP, the mail will be sent to HOST1, then to HOST2, then to HOST3 and finally to USER@HOST4. When source routes are not honored, mail is sent directly to USER@HOST4; the list of hosts is ignored.

If this sample source route is included with a MAIL FROM: command and source routes are honored, the SMTP server will include its host name (for example, `VMHOST1`) in the path information, and will supply the following path for its MAIL FROM: command:

```
<@HOST1,@HOST2,@HOST3,@VMHOST1:USER@HOST4>
```

If such source routes are not honored, the list of hosts is removed from the mail routing path. In addition, the SMTP server will *not* add its host name to the path information.

```
►►──SMSG──server_id──SOURCEroutes──┬─┬─RCPTTO───┬──┬─YES─┬─┬──────────►◄
                                   │ ├─MAILFROM─┤  └─NO──┘ │
                                   │ └─BOTH─────┘          │
                                   └─QUERY─────────────────┘
```

Operands

server_id
: The user ID of the SMTP server virtual machine.

YES
: Indicates that client-supplied source routes on the MAIL FROM: command, the RCPT TO: command, or both will be honored when mail is forwarded by the SMTP server. The **RCPT TO**, **MAIL FROM** or **BOTH** parameter determines the specific source routes honored by the SMTP server.

NO
: Indicates that client-supplied source routes on the MAIL FROM: command, the RCPT TO: command, or both are not to be honored when mail is forwarded by the SMTP server. The **RCPT TO**, **MAIL FROM** or **BOTH** parameter determines the specific source routes ignored by the SMTP server.

Note: The **NO** parameter does *not* cause mail containing source routes to be rejected.

RCPTTO
: Indicates the processing of source routes supplied with the RCPT TO: command is to be affected. When this parameter is used and **NO** is specified, any host list will be ignored and only the destination host will be used. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, the SMTP server will send the mail directly to USER@HOST4; the HOST1, HOST2 and HOST3 hosts will be ignored.

When used with the **YES** parameter, source routes will be honored.

MAILFROM
: Indicates the processing of source routes supplied with the MAIL FROM: command is to be affected. When this parameter is used and **NO** is specified, the list of hosts will be removed from the mail routing path. In addition, the SMTP server will *not* add its host name to the path information. The mail recipient(s) will not see the host list that was ignored.

For the previous sample source route, the SMTP server will supply the following MAIL FROM: command when mail is forwarded:

```
MAILFROM: <USER@HOST4>
```

When used with the **YES** parameter, source routes will be honored, with the SMTP server host name included as part of the path information.

BOTH
: Indicates the handling of source routes supplied for both the RCPT TO: or the MAIL FROM: commands is to be affected. Source routing information for these commands will be processed as previously described.

QUERY
Displays the current SOURCEROUTES setting. The returned response indicates whether or not source routes will be honored.

Examples

- The command that follows will disable source routes for the RCPT TO: command. Source routes will not be honored and will be ignored.

  ```
  smsg smtp sourceroutes rcptto no
  ```

  The following response is displayed:

  ```
  * From SMTP: RCPTTO SOURCEROUTES is set to NO
  ```

Privileged User SMSG TRACE Command

```
►►──SMSG──server_id──TRACE──┬─────────────┬──────────────────────────────────►◄
                            ├─ALL─────────┤
                            ├─CONN────────┤
                            ├─NOCONN──────┤
                            ├─DEBUG───────┤
                            ├─NODEBUG─────┤
                            ├─NOTICE──────┤
                            ├─NONOTICE────┤
                            ├─RESOLVER────┤
                            ├─NORESOLVER──┤
                            ├─CODEFLOW────┤
                            ├─NOCODEFLOW──┤
                            ├─SPL─────────┤
                            ├─NOSPL───────┤
                            └─END─────────┘
```

Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
The user ID of the SMTP server virtual machine.

ALL
Initiates tracing of all types.

CONN
Initiates tracing of connection activity.

NOCONN
Terminates tracing of connection activity.

DEBUG
Initiates tracing of all commands and replies and their associated connection numbers (this is the same information that was captured using the old DEBUG configuration option).

NODEBUG
Terminates tracing of commands and replies.

NOTICE
Initiates tracing of all TCP/IP notification events that are received by the SMTP server.

NONOTICE
Terminates tracing of TCP/IP notification events.

RESOLVER
Initiates resolver tracing. This is the same as adding the **TRACE RESOLVER** statement to the TCPIP DATA file that is read by the SMTP server at initialization.

NORESOLVER
Terminates resolver tracing.

CODEFLOW
Initiates tracing of SMTP program code flow.

NOCODEFLOW
Terminates code flow tracing.

SPL
Initiates tracing of *SPL IUCV execution.

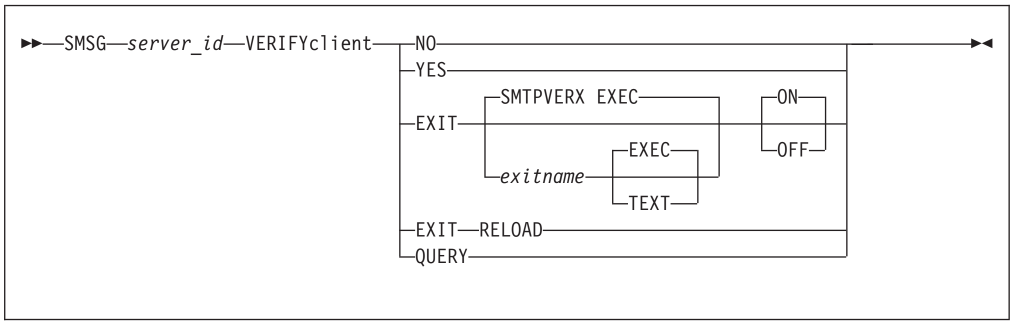
NOSPL
Terminates *SPL IUCV tracing.

END
Terminates all tracing activity.

Note: An SMSG TRACE command entered with no additional parameters is interpreted as a request to report (or query) the tracing activities that are currently in effect.

Privileged User SMSG VERIFYCLIENT Command

The VERIFYCLIENT command is used to indicate whether or not client verification is to be performed, and to query VERIFYCLIENT settings. Client verification can be performed using the built-in client verification function (VERIFYCLIENT YES), or using a user exit (VERIFYCLIENT EXIT). For information on using the client verification exit, see the *TCP/IP Programmer's Reference*.



Privileged user SMSG commands are accepted only from users specified in the SMSGAUTHLIST configuration statement.

Operands

server_id
The user ID of the SMTP server virtual machine.

NO
Indicates no client verification is to be performed.

YES
Indicates verification of the client name specified on the HELO or EHLO command is to be performed using the built-in client verification function.

EXIT
Indicates a client verification exit routine is being turned on or off by this command.

exitname
The name of the exit routine associated with this command (the default exit routine name is SMTPVERX).

EXEC
Indicates the exit routine name specified on this command is the name of an EXEC (this is the default).

TEXT
Indicates the exit routine name specified on this command is the name of a text deck.

ON
Indicates the specified exit is being enabled (turned on).

OFF
Indicates the specified exit is being disabled (turned off).

RELOAD
Forces the exit routine to be reloaded the next time it is executed. If the exit routine is a REXX exec, then it will be EXECLOADed into storage the next time it is executed.

QUERY
Returns current VERIFYCLIENT settings; the returned response indicates whether client verification is enabled (on) or disabled (off).. If an exit has been defined, the name of the exit is included in the response. Sample responses for several settings are shown in Table 29.

Table 29. Client Verification Exit - Sample Queries

Client Verification Setting	Response from Query
Built-in function ON	`* From SMTP: VERIFYCLIENT is set to YES`
Built-in function OFF	`* From SMTP: VERIFYCLIENT is set to NO`
Exit SMTPVERX TEXT enabled	`* From SMTP: VERIFYCLIENT exit SMTPVERX TEXT ON`
Exit SMTPVERX TEXT disabled	`* From SMTP: VERIFYCLIENT is set to NO (Exit SMTPVERX TEXT OFF)`

Examples

- The command that follows will enable the client verification exit routine SMTPVERX EXEC; client verification will be performed by this exit routine.

```
smsg smtp verifyclient exit smtpverx exec on
```

The following response is displayed:

```
* From SMTP: VERIFYCLIENT exit SMTPVERX EXEC ON
```

- This next command will disable all client verification; no client verification is performed.

  ```
  smsg smtp verifyclient no
  ```

  The following response is displayed:

  ```
  * From SMTP: VERIFYCLIENT is set to NO
  ```

- This last command will query the current client verification setting.

  ```
  smsg smtp verifyclient query
  ```

  The following response is displayed:

  ```
  * From SMTP: VERIFYCLIENT is set to NO (Exit SMTPVERX EXEC OFF)
  ```

Chapter 21. Configuring the SNALINK Server

Figure 19 shows the SNALNKA virtual machine interfaces between the TCPIP virtual machine's SNAIUCV driver and the customer's SNA network. The SNALINK application running in the SNALNKA virtual machine communicates with one or more instances of SNALINK at remote nodes, using the SNA LU type 0 protocol.

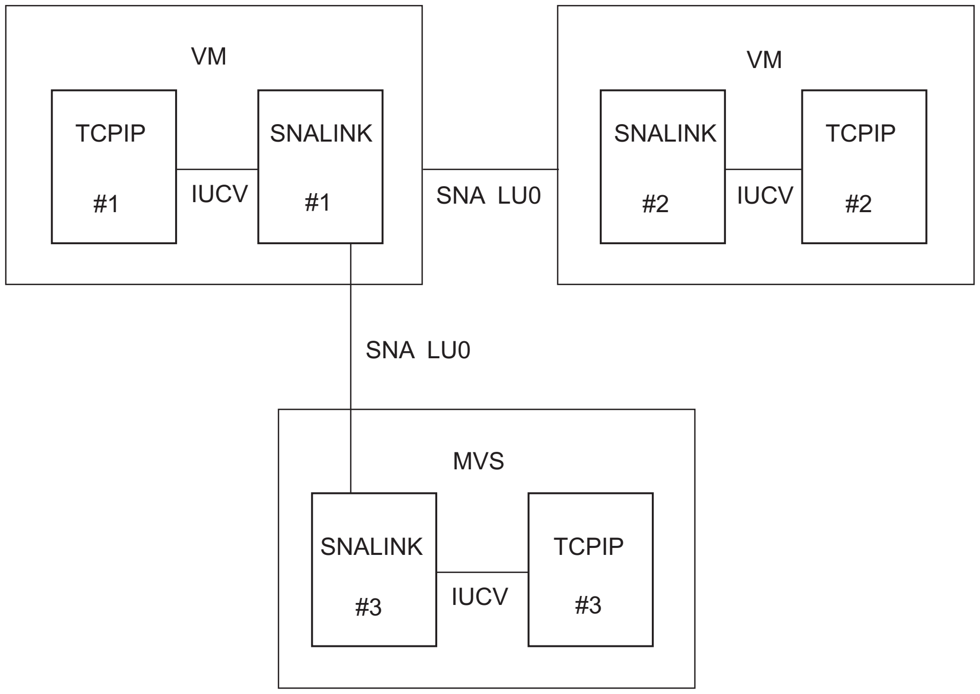


Figure 19. SNALINK Communications

Each SNALINK virtual machine can communicate with up to six other SNALINK machines simultaneously.

Install the SNALNKA Virtual Machine

After defining the SNALNKA virtual machine:

1. Give SNALNKA access to Group Control System (GCS).
2. Define the SNALINK application in the VM/VTAM definition files.
3. Customize the SNALNKA GCS file.

Notes:

1. In the TCPIP and SNALNKA virtual machine directory entries, MAXCONN should be greater than MaxSession.
2. If there are more than two SNA sessions, increase virtual machine storage by 130 Kb over and above the default storage size of 2Mb.

Step 1: Give SNALNKA Access to GCS

The SNALINK application must run in an authorized member of the GCS group. The SNALNKA user ID can be authorized in one of two ways:

1. Re-build GCS, with SNALNKA defined on an AUTHUSER statement in the GCS configuration file, or by using the GROUP exec.
2. The CONFIG AUTHUSER ADD SNALNKA command may be issued from the GCS recovery machine console or profile.

Authorization to IPL the GCS named saved system is required. The CP directory entry for SNALNKA must have NAMESAVE GCS in it.

Step 2: Define SNALINK Applications in VM/VTAM

When operating in dual mode, SNALINK opens two SNA sessions for each remote logical unit (LU) with which it communicates, one for sending and one for receiving. You may have to specify pacing values. Consult your VTAM network administrator for further details. When operating in single mode, SNALINK opens one duplex session.

SNALINK provides its own BIND parameters, so it does not assume or require any particular LOGMODE entries.

In the VTAM APPL definition, the EAS value should be two times the number of MaxSession. Specify AUTHEXIT=YES rather than NO.

SNALINK Installation Considerations

Figure 20 is an example of a typical APPL statement for SNALINK.

```
VM2LUS  APPL ACBNAME=VM2LUS,                                    X
             AUTH=(ACQ,VPACE),                                  X
             AUTHEXIT=YES,                                      X
             EAS=12,                                            X
             PARSESS=YES,                                       X
             SONSCIP=YES,                                       X
             VPACING=0
```

Figure 20. APPL Statement for SNALINK

Step 3: Customize SNALNKA GCS

Figure 21 on page 507 shows the content of a sample SNALNKA GCS file (supplied on the TCPMAINT 591 as file SNALNKA SAMPGCS). Your customized file should be stored on the TCPMAINT 198 disk as file SNALNKA GCS. Within your customized file, update the *TcpipUserid*, *LocalLuName*, and *MaxRuCode* variable assignments to reflect appropriate values for your environment. In addition, add a `CP SPOOL CONSOLE` statement to your SNALNKA GCS file if a user ID other than TCPMAINT is to receive the console log. If additional SNALink servers are required, copy the SNALNKA GCS file to TCPMAINT 198, appropriately naming each copied file as *userid* GCS.

```
/**/
TcpUserid   = 'TCPIP'       /* Userid of TCP/IP stack          */
LocalLuName = 'VM1LU'       /* Must be defined to VTAM         */
MaxRuCode   = 'C7'          /* X'C7' = (12 x 2**7) = 1536 bytes*/
MaxSession  = '6'
Retry       = '0015'        /* Expressed in HHMM format        */
ConnectMode ='SINGLE'       /* Use SINGLE for 3745 NCST session*/

'GLOBAL LOADLIB SNALINK'
'LOADCMD SNALINK SNALINK'
'SNALINK' TcpUserid LocalLuName MaxRuCode MaxSession Retry ConnectMode
Exit rc
```

Figure 21. Sample SNALNKA GCS File

Parameter	**Description**
DEBUG	DEBUG is a prefix parameter and must be the first parameter in the list, if specified. DEBUG enables detailed tracing into an internal buffer. The DEBUG parameter should be specified only if the user is instructed to do so by the Support Center since there is no direct access to the trace table that is created. (The table is a data area internal to the SNALINK executable module and is accessible only by dumping the appropriate portion of the virtual storage of the SNALNKA virtual machine).
TcpipUserid	Identifies, to GCS, the user ID assigned to the TCPIP virtual machine. This variable should be changed only if your TCPIP virtual machine has been named something other than TCPIP.
LocalLuName	Identifies, to GCS, the LU name used on the ACBNAME parameter of the APPL statement in the VTAM definition in Figure 20 on page 506.
MaxRuCode	Specifies the maximum request or response unit (RU) in hexadecimal. This parameter is optional. Set MaxRuCode to specify the maximum RU size that SNALINK sends. The value is of the formX'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. For more information about this parameter, see the*VTAM Programming* manual.
MaxSession	Specifies the maximum number of sessions; a decimal value from 1 to 9999. The default value is 6. To use different values for MaxSession, you also have to specify the MaxRuCode.
Retry	Specifies the delay time for VTAM to retry sense codes and 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, specify both MaxRuSize and MaxSession.

ConnectMode Defines the SNALINK communication session mode. The ConnectMode can have the values of SINGLE, DUAL, or be omitted. If the parameter is omitted, the ConnectMode defaults to DUAL. If the ConnectMode is set to SINGLE, SNALINK creates a single duplex session. If DUAL is specified, SNALINK creates two sessions, a send session and a receive session. Like MaxSession and Retry, if ConnectMode is to be specified you must specify the previous parameters.

SINGLE must be used for NCST sessions in the Network Control Program (NCP) for 3745-type communication controllers.

Operating SNALINK

SNALINK is not started automatically when TCP/IP for VM is installed. Like other applications that run in the GCS environment, the SNALNKA virtual machine needs to be autologged by an appropriate resource, as is done for other machines in the GCS group, such as the VTAM or RSCS machines.

To stop SNALINK, enter the following on the SNALNKA virtual console:

```
REPLY 0 X
```

Restart a Session

If necessary, you can immediately retry a session that is waiting for the retry delay to expire by stopping and starting the connection using the OBEYFILE command.

Sample Console

The example shown in Figure 22 and the accompanying information illustrate SNALink operation. The line number notations have been added for clarity; they do not appear in the console output.

```
        Ready;
Line 1   S (595) R/O
Line 2   A (191) R/W
Line 3   C (591) R/O
Line 4   92275 173617          │ Using a SINGLE LU0 session
Line 5   92275 173617          │ Init complete, APPLID VM2LUS, TCPIP id TCPIP2
Line 6   92275 173617          │ Maximum RU size is 00008000
Line 7   92275 173617          │ Max Sessions 6 : Retry Interval 0030
Line 8   00 REPLY X TO SHUT DOWN
Line 9   92275 173620 VM3LUS   │ IUCV path 00000001 pending
Line 10  92275 173620 VM3LUS   │ Sending SNA BIND request
Line 11  92275 173621 VM3LUS   │ SNA session established
Line 12  92275 173621 VM3LUS   │ Accepting IUCV path 00000001
Line 13  r 0 x                 │ Shutdown requested
         Ready;
Line 14  92275 173629          │ Received WTOR reply, shutting down
Line 15  92275 173629 VM3LUS   │ RECEIVE  CHECK err. R15 00000004 R0 00000010 RTNCD 00000010 FDBK2 0000000D
Line 15  92275 173629 VM3LUS   │ RECEIVE  sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 16  92275 173629 VM3LUS   │ RECEIVE shows RESPOND 00000000 CONTROL 00800000 RH 00000000
         Ready;
```

Figure 22. Sample Console File

The following is the description of the Sample Console File.

Line Number	Description
Lines 1, 2, and 3:	Specifies the minidisk connections with access rights for SNALINK operations.
Lines 4 through 7:	Displays startup information from the command line parameters, which are customized in the sample console. The maximum RU size is displayed in hexadecimal, as are all other numbers except for the time stamp. The time stamp on each message is in the form YYDDD HHMMSS.
Line 8:	Issues a WTOR macro at startup. A reply is interpreted as a request to shut down.
Line 9:	Specifies the TCPIP virtual machine, `TCPIP2`, issues an IUCV CONNECT to establish a session with the remote LU `VM3LUS`. Remote LU `VM3LUS` is higher in collating sequence than the local LU name `VM2LUS`, so SNALINK takes the *passive* role in connecting to `VM3LUS`. This means that it waits for `VM3LUS` to establish a session.
Line 10:	Sends a BIND request to `VM3LUS`.
Line 11:	Indicates that the BIND request was accepted and SNALINK has established the SNA session.
Lines 12:	Establishment of the SNA session causes SNALINK to accept the corresponding IUCV path.
Line 13:	Indicates that the operator has issued a reply to the WTOR, causing SNALINK to shut down. All IUCV paths and SNA sessions are terminated.
Lines 14 through 16:	Indicates that VM3LUS has terminated its sessions, which results in various error messages.

Determine SNA Session Status by SNA IUCV Device Status

The IUCV 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`	Last session was terminated, or last session attempt failed. SNAIUCV driver retries the connection within 30 seconds.
`Connected`	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`	As in `Connected`. In addition, there is an IUCV SEND currently outstanding.

Define SNA IUCV Links

SNA IUCV links are point-to-point, unlike any other supported device.

The IUCV link constitutes a separate network, even though the IUCV link includes only two hosts. You have to assign a unique network or subnetwork number to each end of the IUCV link. The IUCV link does not need its own network or subnetwork number if the link is connecting two hosts that have unique network or subnetwork numbers attached to them and you are not running ROUTED or SNMP.

For example, consider the following scenario:

- Hosts A and B are connected by SNA IUCV
- Host A is also connected to a Token-Ring whose address is 193.1.1
- Host B is also connected to a Token-Ring whose address is 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

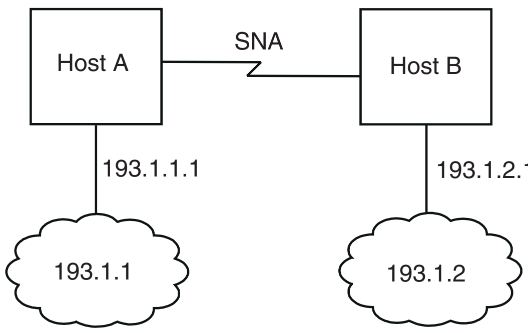


Figure 23. SNA Type 0 IUCV Links

Host A's PROFILE TCPIP could contain:

```
home
    193.1.1.1   tr1
    193.1.1.1   snaiucv1
gateway
*  Network          First hop   Link     Packet size    Subnet mask
   193.1.1          =           tr1          2000        0
   193.1.2          =           snaiucv1     2000        0
```

Host B's PROFILE TCPIP could contain:

```
home
    193.1.2.1   tr1
    193.1.2.1   snaiucv1
gateway
*  Network          First hop   Link     Packet size    Subnet mask
   193.1.2          =           tr1          2000        0
   193.1.1          =           snaiucv1     2000        0
```

The SNA IUCV link does not have its own home address. Hosts A and B are addressed by their Token-Ring home addresses, even if the packets reach them through the SNA IUCV link.

If host B had no other network attached to it, you would have to assign a separate (sub)network number to the SNA IUCV 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.

Chapter 22. Configuring the SNMP Servers

This chapter describes how to configure the Simple Network Management Protocol (SNMP) virtual machines. To monitor a TCP/IP VM network from your VM system, you must use NetView® for z/VM.

SNMP Overview

SNMP is an architecture that allows you to manage an internet environment. You can use SNMP to manage elements such as gateways, routers, and hosts on the network. Network management stations act as clients to run the management applications that monitor the network. Network elements act as servers and contain management agents, which perform the management functions required. SNMP provides the communication between these elements and stations to send and receive information about an internet's resources.

The MIB consists of information about these resources and elements. The MIB is referred to as a database, but it actually exists as counters and temporary storage areas on most of the servers. Data in the MIB is designed according to international standards for internet management and defined according to the International Standard Organization (ISO) Abstract Syntax Notation 1 (ASN.1). IBM has extended the standard MIB by defining enterprise-specific MIB variables for the 3172 Interconnect Controller. For more information about the MIB, see RFC 1155.

More functions are available by using the SNMP daemon Distributed Program Interface (DPI). The DPI permits end users to dynamically add, delete, or replace management variables in the MIB without requiring the recompile of the local SNMP agent. DPI also allows you to generate customized TRAPs. For more information about the SNMP DPI, see *TCP/IP Programmer's Reference*.

TCP/IP network management operates at the application level from one or more hosts within an internet. Each participating host or gateway runs a server program to support the SNMP functions. A NetView operator acting as a network manager invokes client software at the local host computer, which has a specified client server. The client contacts a command processor or query engine and sends queries to obtain information or send commands to vary conditions for a particular managed entity such as a gateway or host TCP/IP link. Only authenticated managers can participate in this process. Figure 24 on page 514 illustrates the VM implementation of SNMP with NetView.

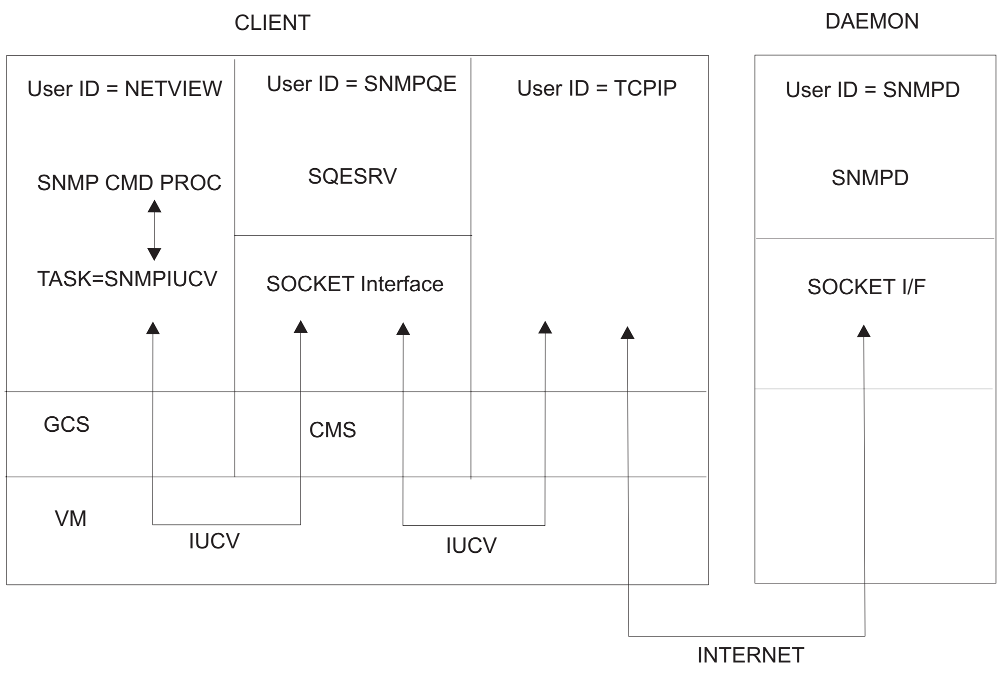


Figure 24. Overview of NetView SNMP Support

The following steps describe how the SNMP command is processed.

1. The NetView operator or CLIST issues an SNMP command.
2. The SNMP command is validated by the SNMP Command Processor.
3. The Command Processor passes the request to the SNMPIUCV task.
4. The SNMPIUCV task passes the request to the SNMP Query Engine.
5. The SNMP Query Engine validates the request, converts the MIB variable to ASN.1 format if necessary, builds the SNMP request and sends it to the SNMP daemon.
6. The SNMP Query Engine receives a response from the SNMP daemon.
7. The SNMP Query Engine decodes the response and sends it to the NetView SNMPIUCV task.
8. The SNMPIUCV task sends the response as a multi-line message to the requesting operator or authorized receiver.

|

SNMP Configuration Steps

| 1. Update the TCPIP server configuration file.
| 2. Update the DTCPARMS file for SNMPQE and SNMPD.

Step 1: Update PROFILE TCPIP

Include SNMPQE and SNMPD in the AUTOLOG statement to automatically start the SNMPQE and SNMPD virtual machines when TCPIP is invoked. Verify that the following statements have been added to PROFILE TCPIP.

```
AUTOLOG
  SNMPQE 0                    ; SNMP Query Engine
  SNMPD 0                     ; SNMP daemon
```

SNMP requires that port UDP 161 be reserved for all messages sent to the VM agent, and that port UDP 162 be reserved for SNMP messages that report TRAPs. Verify that the following statements have been added to PROFILE TCPIP.

```
PORT
  161 UDP SNMPD              ; SNMP daemon port for SNMP messages
  162 UDP SNMPQE             ; SNMP Client port for receipt of TRAPs
```

The SNMP Query Engine and agent use raw sockets for the DPI interface and the SNMP PING function. To allow the SNMP Query Engine (SNMPQE) and the SNMP daemon (SNMPD) to create RAW sockets, add SNMPQE and SNMPD to the OBEY list in the PROFILE TCPIP file. Verify that the following statements have been added to PROFILE TCPIP.

```
OBEY
  SNMPQE SNMPD
```

For more information on the OBEY list, see "OBEY Statement" on page 115.

The MIB II variable `sysContac` identifies the contact person for this managed node. Define the contact person for this node using the SYSCONTACT statement, as shown in the following example:

```
SYSCONTACT
  Andrew Ford, extension 1234
ENDSYSCONTACT
```

The MIB II variable `sysLocation` identifies the physical location of this managed node. Define the physical location of this node using the SYSLOCATION statement, as shown in the following example.

```
SYSLOCATION
  690 Market street, bldg 100
  3rd floor, room 398
ENDSYSLOCATION
```

If IBM enterprise specific variables are to be retrieved from an attached 3172, add the optional NETMAN keyword to the appropriate DEVICE statement as shown in the following example.

```
device lcs1 lcs 9e0 netman
```

Step 2: Update the DTCPARMS File for SNMPQE and SNMPD

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

```
  :Nick.SNMPQE
    :Parms.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the SNMPQE and SNMPD servers if you:

- Activate and specify the level of tracing needed.
- Require a trace of IUCV communication to be done.

SNMP Query Engine

The SNMP Query Engine is a separate VM user ID that runs the SQESERV MODULE. The Query Engine requires access to the MIB_DESC DATA file, which contains the MIB variable descriptions. A sample MIB data file is supplied on the TCPMAINT 591 disk as MIB_DESC SDATA. Your customized file should be stored on the TCPMAINT 198 disk as file MIB_DESC DATA.

```
►►──SQESERV──┬──────────────────┬──┬───────────────┬──┬─────┬──────────────►◄
             └─-d── trace_level─┘  └─-h── hostname─┘  └─-it─┘
```

Specify SQESERV operands as **:Parms.** tag startup parameters in your DTCPARMS file.

Operands

-d *trace_level*
Specifies the level of tracing to be run. Valid values for the trace level are:

0	No tracing (default)
1	Display errors
2	In addition to errors, also displays SNMP Query Engine protocol packets sent and received
3	In addition to SNMPQE packets and errors, also displays the SNMP packets sent and received

-h *hostname*
Specifies the IP address to bind to, so that SQESERV accepts only connections through that IP address. This parameter is useful if multiple IP addresses exist for a single host and you want to restrict access from one side.

-*it* Specifies that a trace of IUCV communication is to be done. Used for debugging only the socket layer in a user's application. It can result in a very large amount of output.

Step 3: Define the MIB Data File

The Management Information Base (MIB) data file, MIB_DESC DATA, defines the short names for MIB variables. Short names are the character representation for the ASN.1 variable names. For example, `sysUpTime` is the short name for 1.3.6.1.2.1.1.3.0 (the MIB variable that stores the time since the MIB object was last restarted). Short names are generally shown as a combination of upper and lowercase characters, though SNMP for VM ignores these case distinctions. Variable names must always be in ASN.1 language when they are sent to an SNMP daemon. You can always use ASN.1 language to specify the variable names in an enterprise-specific tree (assuming that the agent supports them). You can use these short names to specify the MIB variables.

When you issue an SNMP GET, GETNEXT, or SET command, and specify the variable name in ASN.1 notation, the SNMP Query Engine uses that name and sends it in the SNMP packet to the agent. When you issue an SNMP GET, GETNEXT, or SET command, and specify the short name for the variable (for example, `sysDescr`), the SNMP Query Engine looks for that name in the MIB_DESC DATA file and uses the ASN.1 name specified in the file when it sends the SNMP packet to the agent.

The distributed MIB_DESC DATA file contains the text names as defined in RFC 1156. In addition to these, the following variables have been added:

- MIB-II variables from RFC 1158
- IBM 3172 Enterprise Specific MIB variables and some IBM-defined variables in the enterprise-specific branch of the MIB

The SNMPQE virtual machine must be able to access the MIB_DESC DATA file.

You can change the short names in the MIB_DESC DATA file to the equivalent in your national language. You can also leave the current names and add the equivalent names in your national language. However, the SNMP MIBVNAME function returns only the first entry found in the file that satisfies the search. In addition, all enterprise-specific variables used by hosts in your network should be added to this file.

Entries in the file do not need to be in a specific sequence. Each name starts on a new line. The following shows the line format.

```
short_name asn.1_name type time_to_live
```

Each variable on the line is separated by either one or more spaces or tabs. An asterisk (*) in column 1 indicates that the line is a comment line.

Figure 25 is an example of an MIB_DESC DATA line with a `sysDescr` variable translated in Dutch and a few enterprise variables added (in this example, company ABC received 1.3.6.1.4.1.42 as the ASN.1 number for their enterprise). A sample MIB_DESC DATA file is shipped on TCPMAINT 591. It should be copied to TCPMAINT 198 and customized.

```
*-------------------------------------------------------------*
* MIB Variable name | ASN.1 notation        | Type     | TTL  *
*-------------------------------------------------------------*
sysDescr            1.3.6.1.2.1.1.1.        display    900
* Following is Dutch name for sysDescr
systeemBeschrijving  1.3.6.1.2.1.1.1.        display    900
  ...
  other entries
  ...
* Following are enterprise specific variables for company ABC
ABCInfoPhone        1.3.6.1.4.1.42.1.1      display    900
ABCInfoAddress      1.3.6.1.4.1.42.1.2      display    900
```

Figure 25. Sample MIB_DESC DATA Line

The TTL field contains the number of seconds that a variable lives in the Query Engine's internal cache. If there are multiple requests for the same variable within the TTL period, the variable value is obtained from the cache, and unnecessary network traffic is avoided.

You can define multiple short names or text names for the same variable, as shown with the Dutch translation of the sysDescr variable. In this case, the SNMP Query Engine returns the first value in the table on an SNMP MIBVNAME request. In

Figure 25 on page 517, the SNMP Query Engine would return `sysDescr` and not `systeemBeschrijving`. 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 MIB_DESC DATA file. This comparison is not case-sensitive. For example, a request for `SYSDESCR`, `SysDescr`, or `sysDescr` matches the `sysDescr` entry with an ASN.1 notation of `1.3.6.1.2.1.1.1.`.

When the SNMP Query Engine receives an SNMP response, it looks up the variable in the MIB_DESC DATA table Type field for information about translating the value into displayable characters. The information contained in the Type field is case-sensitive and must be specified in lowercase.

Step 4: Configure the SNMP/NetView Interface

This section describes how to configure the SNMP/NetView Interface.

SNMPIUCV

SNMPIUCV is the NetView optional task that handles IUCV communication with the SNMP Query Engine. You need to add the following TASK statement for SNMPIUCV to the DSIDMN NCCFLST file.

```
TASK  MOD=SNMPIUCV,TSKID=SNMPIUCV,PRI=5,INIT=Y
```

This statement causes SNMPIUCV to start automatically when NetView is started. If you use `INIT=N` in the TASK statement for SNMPIUCV, a NetView operator can start the SNMPIUCV task by entering the following:

```
START TASK=SNMPIUCV
```

The SNMPIUCV task tries to connect through IUCV to the SNMP Query Engine. If this fails, it retries the connect as specified in the SNMPARMS NCCFLST file, see “SNMPIUCV Initialization Parameters” on page 519, the default is every 60 seconds.

SNMP Command Processor

SNMP is the command processor that allows NetView operators and CLISTs to issue SNMP commands. You should add the following statement to the DSICMD NCCFLST file.

```
SNMP  CMDMDL  MOD=SNMP,ECHO=Y,TYPE=R,RES=N
```

`RES=Y` to make the command resident in memory to improve performance.

After the SNMPIUCV task is started, you can issue the SNMP command. The SNMP command passes a request to the SNMPIUCV task to forward to SNMPQE. The return code represents a request number that is associated with the request. The responses are returned asynchronously and contain this request number. The operator or CLIST must use the request number to correlate the response to the request.

SNMP Messages

The SNMP messages reside in the DSISNM*nn* NCCFLST files, which are shipped on the server code disk, TCPMAINT 591 (*nn* indicates the number of the message).

The valid message files are DSISNM00 through DSISNM05, DSISNM10, DSISNM12, and DSISNM99. These files should reside on a disk that the NetView user ID can access.

SNMPIUCV Initialization Parameters

SNMPIUCV reads the SNMPARMS NCCFLST file at startup. This file contains the initialization parameters for SNMP and is shipped on the server code disk, TCPMAINT 591. You should place the SNMPARMS NCCFLST file on any disk that can be accessed by the NetView virtual machine.

The following example shows the parameters and the default values of the SNMPARMS NCCFLST file.

```
*
*      SNMPARMS NCCFLST
*
  SNMPQE   SNMPQE   * Userid of SNMP Query Engine
  SNMPQERT 60       * Retry timer (seconds) for IUCV CONNECT
  SNMPRCNT 2        * Retry count for sending SNMP requests
  SNMPRITO 10       * Retry initial timeout (10ths of a second)
  SNMPRETO 2        * Retry backoff exponent (1=linear, 2=exponential)
  SNMPMMLL 80       * Line length for Multiline Messages 38/44
```

You can change the parameters in the SNMPARMS NCCFLST file. The following describes each of the keywords and parameters:

Parameter	Description
SNMPQE *name*	Specifies the virtual machine of the SNMP Query Engine. The default is SNMPQE. If you change the name of the SNMP Query Engine virtual machine to something other than SNMPQE, you must also change this parameter to match. This value is case sensitive.
SNMPQERT *seconds*	Specifies the retry timer (seconds) for IUCV CONNECT. When SNMPIUCV is started, it tries to connect to the SNMP Query Engine. If the connection fails or breaks, SNMPIUCV retries a connect every *n* seconds, as specified by this parameter. The valid range of values is 0 to 9999. The default is 60 seconds.
SNMPRCNT *number*	Indicates the retry count for sending SNMP requests. This is the number of times the SNMP Query Engine resends an SNMP PDU when no response was received. If no response was received after all retries have been exhausted, the SNMP Query Engine returns a no response error for the SNMP request. The valid range of values is 0 to 255. The default is 2. If the request being sent by the SNMP Query Engine contains an invalid community name, no response is received. This causes the SNMP Query Engine to resend the request until the retry count is exhausted. The agent generates multiple authenticationFailure traps, one for the initial request and one for each retry.
SNMPRITO *tenths_seconds*	Specifies the time-out value for the request in

	tenths of a second. After sending an SNMP request to an agent, the SNMP Query Engine waits the specified time for a reply. If no reply is received within the specified time limit, the SNMP Query Engine resends the request the number of times specified by SNMPRCNT. If no replies have been received after all retries have been exhausted, the SNMP Query Engine returns a NO RESPONSE error to NetView. The valid range of values is 0 to 255. The default is 10 tenths of a second.
SNMPRETO *exp*	Indicates the retry back-off exponent. Specifies whether the time-out value between retries of an SNMP request is calculated linearly or exponentially. The valid values are 1 (linear) or 2 (exponential). The default is 2. For example, if the retry time-out was 1 second, SNMPRETO of 1 causes a new retry to be sent at constant one second intervals until all retries have been sent. SNMPRETO of 2 causes the first retry to be sent after one second, the second retry 2 seconds later, the third retry 4 seconds later, and so on until all retries have been sent.
SNMPMMLL *length*	Indicates the line length for multiline messages 38 through 44. The maximum length is 255. A value of 80 allows the complete text to appear on an 80 character-wide screen. The default length is 80 characters.

Configure the SNMP Daemon

This section describes how to configure the SNMP daemon.

```
►►──SNMPD──┬────────────────┬──────────────────────────────────────────►◄
           └─-d trace_level─┘
```

Operands

-d *trace_level*
Specifies the level of tracing to be started. Valid values for *trace_level* are:

0	No tracing (default)
1	Trace snmpd internals
2	Trace external changes from egp
3	Trace incoming requests
4	Trace outgoing responses
5	Trace all levels

TRAP Destination file

TRAPs are unsolicited messages that are sent by an SNMP daemon to an SNMP network management station. An SNMP TRAP contains information about a significant network event. The management application running at the management station interprets the TRAP information sent by the SNMP daemon.

The following TRAPs are generated by an SNMP daemon in TCP/IP.

- COLD_START
- AUTHENTICATION_FAILURE
- LINK_UP
- LINK_DOWN

Note: The SNMP daemon Distributed Program Interface (DPI) allows external processes (which can be running on another host) to generate TRAPs. This can allow for support of other types of TRAPs. For more information about SNMP DPI, see *TCP/IP Programmer's Reference*.

To use TRAPs, create a file called SNMPTRAP DEST. This file defines a list of managers to which TRAPs are sent. The following describes how to set up SNMPTRAP DEST.

1. Create a file called SNMPTRAP DEST. This file must be on a disk accessible to the SNMP agent, such as TCPMAINT 198.
2. The SNMPTRAP DEST file has a list of managers who are to receive the TRAPs, and identifies the protocol used to send TRAPs. The format of an entry in the file is:

   ```
   manager UDP
   ```

   The *manager* is the host that the TRAP is to be sent to. This can be a host name, or it can be the IP address of the host. The protocol must be UDP. There should be one entry in the file for each host to which you want to send traps.
3. A SNMPTRAP DEST file might contain the following:

   ```
   124.34.216.1 UDP
   Host1        UDP
   ```
4. Comments and sequence numbers are not allowed in the SNMPTRAP DEST file.

PW SRC File

SNMP agents are accessed by remote network management stations. These network management stations can be located anywhere in the internet. With the exception of TCP/IP, the network management stations do not have to be directly linked to the host running the SNMP agent.

To allow network management stations to send inquires to the SNMP agent, you must create a file called PW SRC, which defines a list of community names and IP addresses that can use the services of the SNMP agent.

The community names reside in a master file. This file should be created and kept at a secure location, accessible only to a system administrator and the SNMP daemon.

The following describes how to create a community name file.

1. Create the PW SRC file on the TCPMAINT 198 minidisk accessible to the SNMP daemon.

   Note: Since the PW SRC file can contain passwords, you should control access to this file if security is a concern.
2. The PW SRC file has a list of community names that can use the services of the SNMP daemon. The format of an entry in this file is:

   ```
   community_name network_addr network_mask priv_mask
   ```

   The *community_name* can be up to 15 characters in length. This value can contain both upper and lower case letters, however it is case sensitive. In any requests received by the SNMP agent, the *community_name* must match the *community_name* specified in PW SRC exactly, including correct case.

   Note: The *priv_mask* can be used to start or stop tracing of events in the SNMP agent. If *priv_mask* is omitted, the following default mask is assumed:

   ```
   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxs
   ```
3. The following shows a sample PW SRC file containing multiple entries.

```
* This is a sample PW SRC file.  First we see some sample entries.
* NetView on the mainframe passes Uppercase community names from the
* command line, that is why we have the NVTEST entry as a sample.
*
* community_name netw_addr  netw_mask    priv_mask
*
public         0.0.0.0     0.0.0.0
monitor        0.0.0.0     0.0.0.0
NVTEST         0.0.0.0     0.0.0.0
password1      0.0.0.0     0.0.0.0
password2      9.0.0.0     255.0.0.0
password3      9.132.2.4   255.255.255.255
*------------------------------------------------------------
*
* The following community names will turn tracing within the
* agent on/off when a packet arrives with such a community_name.
* This is achieved by specifying an appropriate priv_mask
*
*               /-------> bit 0 (trace on or off)
*               |/------> bit 1 (trace SNMP responses)
*               ||/-----> bit 2 (trace SNMP requests)
*               |||/----> bit 3 (trace external)
*               ||||/---> bit 4 (trace internal)
*               |||||/--> bit 5 (trace DPI)
*               |||||/--> bit 6 (trace DPI internals)
*               ||||||/-> bit 7 (reserved)
*               |||||||                    bit 31 (last bit)
*               vvvvvvv                        v
* Here, string  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxs
* is used by the agent to grant access and turn tracing on/off.
* The string represents bits in a 32 bit integer.
* The last bit (if not set to s) means you have access.
* Bit zero turns tracing on (if set to s) or off (if set to x)
* Bits 1-7 specify which tracing is turned on or off.
*
debug_reply      0.0.0.0  0.0.0.0  ssxxxxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_reply_off  0.0.0.0  0.0.0.0  xsxxxxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_req        0.0.0.0  0.0.0.0  sxsxxxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_req_off    0.0.0.0  0.0.0.0  xxsxxxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_ext        0.0.0.0  0.0.0.0  sxxsxxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_ext_off    0.0.0.0  0.0.0.0  xxxsxxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_int        0.0.0.0  0.0.0.0  sxxxsxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_int_off    0.0.0.0  0.0.0.0  xxxxsxxxxxxxxxxxxxxxxxxxxxxxxxxs
debug_dpi        0.0.0.0  0.0.0.0  sxxxxsxxxxxxxxxxxxxxxxxxxxxxxxxs
```

```
debug_dpi_int    0.0.0.0  0.0.0.0  sxxxxxsxxxxxxxxxxxxxxxxxxxxxxxxs
debug_all_dpi    0.0.0.0  0.0.0.0  sxxxxssxxxxxxxxxxxxxxxxxxxxxxxxs
debug_dpi_off    0.0.0.0  0.0.0.0  xxxxxssxxxxxxxxxxxxxxxxxxxxxxxxs
debug_all        0.0.0.0  0.0.0.0  sssssssxxxxxxxxxxxxxxxxxxxxxxxxs
debug_all_off    0.0.0.0  0.0.0.0  xssssssxxxxxxxxxxxxxxxxxxxxxxxxs
debug_off        0.0.0.0  0.0.0.0  xssssssxxxxxxxxxxxxxxxxxxxxxxxxs
```

4. Comments and sequence numbers are not allowed in the PW SRC file.

The IP address of an incoming SNMP request is ANDed with the first network mask in the PW SRC file. If the result matches the corresponding network address, the community names are compared. If the names match the request is accepted, unless the privilege mask does not specify 's' in the last position. If the network address or community names do not match, the search continues until all entries have been tested.

If a request from IP address 9.34.22.122 with community name password2 is received, IP address 9.34.22.122 is ANDed with 255.0.0.0 because that is the first entry found in the table. The first four entries are skipped (the community name does not match.) The result is 9.0.0.0, which equals the specified network address for `c_name1`. If this incoming request has specified a community name of `password2`, then the request is accepted. In `password3`, only requests from host 9.132.2.4 are accepted, providing that the community names match.

If the IP address ANDed with the network mask does not match a *desired_network*, or if the *community_names* do not match, an `AUTHENTICATION_FAILURE` TRAP is sent, provided that a destination entry exists in the SNMPTRAP DEST file.

A *desired_network* and network mask of all zeros allows anyone with the correct *community_name* to make requests.

Installation Steps

SNMP requires NetView as the client of the local host.

SNMP Command Processor and SNMPIUCV on NetView

This section describes how to install the SNMP Command Processor and SNMPIUCV.

1. If necessary, move the SNMPLIB LOADLIB to the NetView 191 disk or any other disk to which NetView has access, from the server code disk, TCPMAINT 591, where the file was shipped.
2. Add this load library to the GLOBAL LOADLIB statement in the NetView start-up procedure. This file has a file type of GCS. For example, the following is a typical GLOBAL command.

   ```
   GLOBAL LOADLIB STATMON NLDM NPDA PROPMX USER
   ```

   Now change it to:

   ```
   GLOBAL LOADLIB STATMON NLDM NPDA PROPMX SNMPLIB USER
   ```

   As an alternative, you can add the SNMP and SNMPIUCV load modules to the USER LOADLIB.
3. Add a TASK statement to the DSIDMN NCCFLST file.

   ```
   TASK  MOD=SNMPIUCV,TSKID=SNMPIUCV,PRI=5,INIT=Y
   ```

   If you code `INIT=N` on the TASK statement for SNMPIUCV, the NetView operator must start the task manually with the following command.

```
START TASK=SNMPIUCV
```

4. Add the following statement to the DSICMD NCCFLST file.

   ```
   SNMP  CMDMDL  MOD=SNMP,ECHO=Y,TYPE=R,RES=N
   ```

   If you issue many SNMP commands, you can use `RES=Y` to make the command processor resident.
5. If necessary, put all the DSISNM*nn* NCCFLST files on the NetView 191 disk (or a disk accessible to NetView) from the server code disk, TCPMAINT 591, where the files were shipped. These are the externalized messages.
6. If necessary, put the SNMPARMS NCCFLST file on the NetView 191 disk or any other disk to which NetView has access, from the server code disk, TCPMAINT 591, where the file was shipped. These are the SNMP startup parameters. You can change these parameters if the default values are not acceptable. For more information, see "SNMPIUCV Initialization Parameters" on page 519.

SNMP Daemon

This section describes how to install the SNMP agent.

1. Ensure that the TCPIP DATA file and, optionally, the HOSTS ADDRINFO and HOSTS SITEINFO files are on a disk accessible to the SNMPD virtual machine.
2. Ensure that the SNMPTRAP DEST file (see "TRAP Destination file" on page 521), and the PW SRC file (see "PW SRC File" on page 521) are on a disk accessed by SNMPD.
3. Update the DTCPARMS file. For more information, see "Step 2: Update the DTCPARMS File for SNMPQE and SNMPD" on page 515.
4. Add SNMPD to the AUTOLOG, PORT, and OBEY statements in the PROFILE TCPIP file. For more information, see "Step 1: Update PROFILE TCPIP" on page 514.
5. Add the SYSCONTACT and SYSLOCATION statements in the PROFILE TCPIP file. For more information, see "Step 1: Update PROFILE TCPIP" on page 514.
6. If IBM Enterprise-specific MIB variables are to be obtained from an attached 3172, add the optional NETMAN keyword to the DEVICE statement for the 3172.

Chapter 23. Configuring the SSL Server

The Secure Socket Layer (SSL) server provides the processing that allows secure (encrypted) communication between a remote client and a TCP/IP server (in this context, known as the application server). The application server must be listening on a port identified as SECURE by your installation, and the remote client must support the SSL protocol according to RFC 2246.

Under SSL protocol, the application server is always authenticated. To participate in an SSL session, an application server must provide a certificate to prove its identity. Server certificates are issued by Certifying Authorities (CAs), each of which establishes its own identity by providing a CA certificate. Server certificates and CA certificates are stored in a certificate database managed by the SSL server.

No changes to an application server are necessary to participate in SSL. The application server does not perform any data encryption or decryption; this is handled by the SSL server.

The SSL server runs on the Linux operating system, which runs as a guest in the SSL server virtual machine. This Linux operating system is specially configured to support **only** the SSL application; **general use of this Linux operating system is not supported**. No knowledge of Linux is required to configure the SSL server or to perform any SSL server administration functions.

To configure the SSL server, you must perform the following steps:

SSL Server Configuration Steps

1. Update the PROFILE TCPIP file.
2. Update the DTCPARMS file for the SSL server.
3. Update the ETC SERVICES file.
4. Set up the certificate database.

Dynamic Server Operation

The SSL server provides an SSLADMIN command interface that allows you to perform certificate database administration and server administration tasks. See "Dynamic Server Operation" on page 530.

Overview of an SSL Session

An SSL session consists of the following general processing steps:

1. Connect

   A remote client sends a connection request for an application server. If the application server is listening on a secure port, the TCP/IP (stack) server sends this request to the SSL server. The TCP/IP server also sends a label that identifies the certificate to be used for the secure connection, as well as the socket addresses of the client and the application server. The SSL server accepts the connection from the client and sends a connection request to the application server.

The SSL session is maintained as two separate connections: the connection from the remote client to the SSL server, and the connection from the SSL server to the application server. The intervention of the SSL server is transparent to the client and the application server; to them, it seems that they are communicating directly with each other.

2. Handshake

   After its connection request is accepted, the client initiates a handshake protocol to produce the cryptographic parameters for the session. The SSL server (representing the application server) responds to the handshake and sends the application server's certificate to the client.

   Clients that make use of SSL services generally have the certificates associated with the Well Known CAs in their certificate databases. The client compares the "signature" on the application server's certificate with the appropriate CA certificates to verify the authenticity of the server.

   The client and the SSL server then agree on a protocol version, select cryptographic algorithms (known as cipher suites), and use asymmetric (public-key) encryption techniques to generate shared secrets. From the shared secrets, the SSL server and the client generate the symmetric (private) keys to be used for the encryption and decryption of data sent on the connection.
3. Data transmission

   When the handshake completes, the client sends encrypted data over the network. The SSL server receives the encrypted data from the client, decrypts it, and sends it to the application server. The application server responds by sending unencrypted data to the SSL server. The SSL server receives the unencrypted data from the application server, encrypts it, and sends it to the client.
4. Close

   When a close request is received from either the client or the application server, the SSL server sends a close request to the other party and cleans up the connection.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  SSLSERV  0           ;  SSL Server
```

Note: If you do not want the SSL server automatically started, do not include the AUTOLOG statement in the file.

The IBM-supplied PROFILE STCPIP file contains the following statement to reserve TCP port 9999 for the SSL server:

```
PORT
  9999 TCP SSLSERV      ;  SSL Server - administration
```

Verify that a port statement for the SSL server is included in your TCPIP server configuration file and that this port matches the SSL administration port defined in the ETC SERVICES file. See "Step 3: Update ETC SERVICES" on page 529.

Use the OBEY statement to list the privileged users (such as TCPMAINT) who can issue SSLADMIN commands to administer the SSL server. Verify that an OBEY statement is included in your TCPIP server configuration file:

```
OBEY
  TCPMAINT
ENDOBEY
```

Step 2: Update the DTCPARMS File

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

```
:Nick.SSLSERV   :Type.server   :Class.ssl

:Nick.ssl    :Type.class
             :Name.SSL daemon
             :Command.VMSSL
             :Diskwarn.YES
             :Parms.
```

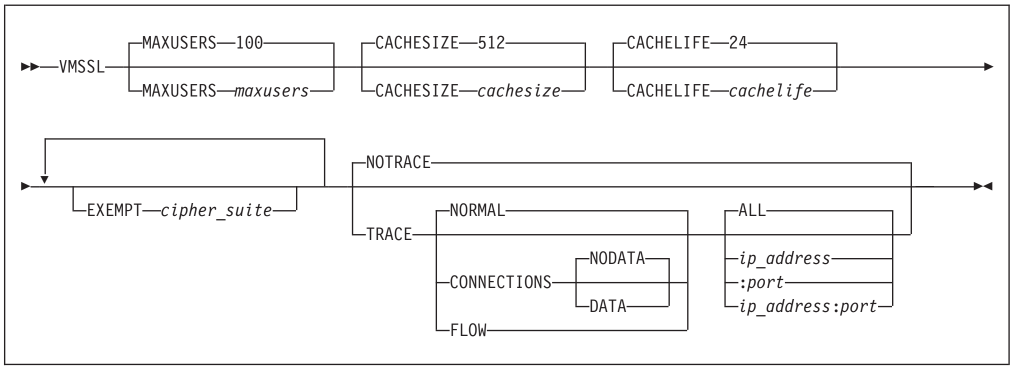
If more customization is needed than what is available in the DTCPARMS file, a server profile exit can be used.

For more information about the DTCPARMS file, customizing servers, and server profile exits, see “Chapter 3. General TCP/IP Server Configuration” on page 17.

Note: You should modify the DTCPARMS file for the SSL server if you want to override the default VMSSL command operands.

VMSSL Command

VMSSL services are initiated using the VMSSL command:



Specify VMSSL command operands as **:Parms** tag start-up parameters in your DTCPARMS file.

Operands

MAXUSERS *maxusers*
specifies the maximum number of concurrent secure sessions to be supported. The default is 100.

CACHESIZE *cachesize*
specifies the maximum number of cache entries the SSL server is to maintain.

Each entry is the result of a handshake agreement between the SSL server and a client. The minimum *cachesize* is 0, the maximum is 4095, and the default is 512.

CACHELIFE *cachelife*
specifies the number of hours that a cache entry is valid, after which the entry expires and can no longer be used. The minimum *cachelife* is 0 and the maximum is 24, which is also the default.

EXEMPT *cipher_suite*
specifies a cipher suite that should **not** be used by the SSL server. The possible values for *cipher_suite* are:

Value	Type of Encryption
RC4_128_MD5	Stream cipher, 128-bit key size, MD5 hash
RC4_128_SHA	Stream cipher, 128-bit key size, SHA hash
DES_56_SHA	Block cipher, 56-bit key size, SHA hash
RC4_56_SHA	Stream cipher, 56-bit key size, SHA hash
RC4_40_MD5	Stream cipher, 40-bit key size, MD5 hash
RC2_40_MD5	Block cipher, 40-bit key size, MD5 hash
DES_40_SHA	Block cipher, 40-bit key size, SHA hash
NULL	No encryption

NOTRACE
specifies that all tracing is turned off. This is the default.

TRACE
specifies that tracing is to be performed.

NORMAL
specifies that a trace entry is recorded to indicate a successful connection. This is the default if TRACE is specified.

CONNECTIONS
specifies that a trace entry is recorded for connection state changes and handshake results.

NODATA
specifies that no data is displayed for send and receive trace entries. This is the default if CONNECTIONS is specified.

DATA
specifies that the first 20 bytes of data are displayed for send and receive trace entries.

FLOW
specifies that flow of control and system activity are traced.

ALL
specifies that tracing is done for all connections. This is the default if TRACE is specified.

ip_address
specifies that tracing is done only for activity associated with this IP address.

:*port*
specifies that tracing is done only for activity associated with this port.

Usage Notes

1. DTCPARMS file changes become effective only when the SSL server is restarted.
2. If the MAXUSERS limit is reached, new connection requests for a secure port are rejected.
3. If the CACHESIZE limit is reached, new entries are not saved in cache.
4. For information about adjusting the CACHESIZE and CACHELIFE settings, see "Using the Server Cache" on page 534.
5. Cipher suite NULL provides no security. Exempting all the cipher suites ***except*** NULL means that no data is encrypted.
6. The SSL server will not start if all the cipher suites are exempted.
7. All IP addresses must be specified in dotted decimal format. If you want to include a port number with an IP address, use the format: *ip_address***:***port*.
8. Certain informational messages are always displayed at the SSL server console for:
   - SSLADMIN commands
   - Potential security breaches, such as a message digest not matching the message during the handshake
9. For information about trace output, see the *z/VM: TCP/IP Level 3A0 Diagnosis Guide*.

Step 3: Update ETC SERVICES

The ETC SERVICES file lists the Well Known Port Numbers as listed in RFC 1700, along with IBM overrides and extensions. The IBM-supplied ETC SERVICES file contains the following statement in the extensions section to define TCP port 9999 for SSL administration:

```
ssladmin      9999/tcp                # administration port
```

Verify that a similar statement is included in your ETC SERVICES file and that the port matches the port reserved for the SSL server in the PROFILE TCPIP file. See "Step 1: Update PROFILE TCPIP" on page 526.

Step 4: Set Up the Certificate Database

The following steps describe how to set up the certificate database. After you complete these steps, you are ready to receive connection requests for your secure servers.

Step 4a: Start the SSL Server

The SSL server must be running and TCPMAINT 591 must be accessed before you can issue SSLADMIN commands. If the server has not been autologged, you can start it manually. See "Starting and Stopping TCP/IP Servers" on page 30.

Starting the SSL server the first time creates the certificate database. The database is preloaded with some Well Known CA certificates. To see what certificates are preloaded, issue the SSLADMIN QUERY command:

```
ssladmin query certificate *
```

Step 4b: Populate the Certificate Database

You need to obtain certificates for the application servers that will be designated as SECURE (listening on secure ports). When an SSL connection is established, the client expects to receive a server certificate during the handshake that authenticates the application server. The server certificate contains information about the application server, such as its fully-qualified name and its public key. The certificate is generally signed by a CA.

Before populating the certificate database with real certificates, you may wish to conduct some tests of the SSL environment. There is a type of certificate called a self-signed certificate that you can create for this purpose. See "Creating a Self-Signed Certificate" on page 533.

To begin populating the certificate database with real certificates:

1. Create a certificate request for each secure server. See "Requesting a Server Certificate" on page 532.
2. When the CA returns the server certificate, store it in the certificate database. See "Storing a Certificate in the Database" on page 532.

Step 4c: Designate the Secure Ports

You must update the TCP/IP server configuration file to designate your secure ports and to associate those ports with the certificates you have stored in the certificate database.

To designate a secure port for a TCP/IP server, you must include the SECURE operand on the PORT statement that reserves that port for the server. For more information, see "PORT Statement" on page 117. When a port is designated as SECURE, a connection request from a client to the server that listens on that port is routed by TCPIP to the SSL server. If SECURE is not specified, the port is considered not secure, and connection requests do not involve SSL processing.

In addition to the SECURE operand, the PORT statement for a secure port must include the label of the server certificate to be used for secure connections to that port.

Note: For testing purposes, you can specify the label of a self-signed certificate instead of a server certificate.

To verify that a certificate with this label resides in the certificate database, issue the SSLADMIN QUERY command:

```
ssladmin query certificate label
```

After you designate the secure ports in the TCP/IP server configuration file, you must activate these changes in one of the following ways:

- To make the changes permanent, you must restart the TCP/IP (stack) server.
- To activate the changes dynamically, use the OBEYFILE command. For more information, see "OBEYFILE Command" on page 135.

Dynamic Server Operation

The SSL server provides an SSLADMIN command interface that allows you to perform certificate database administration and server administration tasks.

Certificate Database Administration

This section describes how to do various certificate database administration tasks, such as requesting a server certificate and storing a certificate in the database. It also describes the contents of the X509INFO file, which provides input for some SSLADMIN commands.

Notes:

1. The SSL server must be running and TCPMAINT 591 must be accessed before you can issue SSLADMIN commands.
2. The SSLADMIN commands work with a copy of the certificate database on disk. Starting the SSL server loads the certificate database from disk into memory. When you make changes to the certificate database, such as storing or deleting a certificate, those changes do not go into effect until you restart the SSL server to load the updated certificate database into memory.

X509INFO File

The *label* X509INFO file is a CMS file that you create to specify information used as input when requesting a certificate for an application server (SSLADMIN REQUEST or SSLADMIN SELF command). The file name of this file will be used as the label that is associated with the certificate request, and later the certificate itself, in the certificate database. You will specify this label on the PORT statements in the TCP/IP configuration file that define the secure ports for this application server.

The format of an X509INFO file is:

```
Common common_name
Organization organization
Unit unit
Locality city
State state
Country country
```

Each keyword and value pair must be specified on a separate line. If the first nonblank character in a line is an asterisk (*), the line is treated as a comment. Blank lines and unidentified keywords are ignored.

Note: A forward slash (/) is not allowed on any line in the X509INFO file. A backward slash (\) is treated as an escape character.

Certain lines in this file are required and others may be optional (as determined by the CA). The lines specify the following information:

Common	(Required) Fully-qualified domain name of the application server.
Organization	(Required) Organization that owns the domain name.
Unit	(Optional) Division within the organization.
Locality	(Optional) City in which the organization is located.
State	(Optional) State in which the organization is located.
Country	(Required) Two-character ISO-format code identifying the country in which the organization is located. If this line is omitted, or if the keyword is specified without a value, the value defaults to US.

Requesting a Server Certificate

An application server cannot participate in an SSL session unless a certificate for the server is in the certificate database. When an SSL session is being established, the SSL server sends the server certificate to the client (on behalf of the application server) during the handshake.

To obtain a server certificate:

1. Create a *label* X509INFO file containing information about the application server.

   See "X509INFO File" on page 531 for information about the structure of this file. Consult with the CA you intend to use about what information they require in the certificate request and for instructions on how to send the certificate request.
2. Issue the SSLADMIN REQUEST command to create the certificate request:
   ```
   ssladmin request label keysize fm
   ```
   For more information, see "SSLADMIN REQUEST Command" on page 541.
3. Send the certificate request to the CA.

Storing a Certificate in the Database

In response to a certificate request, the CA returns a server certificate that you must store in the certificate database. The CA may also send one or more CA certificates. You must store the CA certificates in the certificate database before you store the server certificate, because the CA certificates are used to verify the server certificate. If the CA returns only one certificate, it may be a server certificate or it may be a certificate chain consisting of a server certificate and one or more CA certificates.

To store a certificate in the certificate database:

1. Receive the certificate into a CMS file with a file type of X509CERT.

   If the CA returns more than one certificate, receive each certificate into a distinct X509CERT file.
2. Issue the SSLADMIN STORE command to store the certificate in the certificate database, specifying the file name of the X509CERT file that contains the certificate to be stored:
   - If you have received any separate CA certificates, issue the SSLADMIN STORE command with the CA operand for each CA certificate, giving each CA certificate a unique label:
     ```
     ssladmin store fn ca label
     ```
     Note: The label is case-sensitive and cannot exceed 200 characters.
   - After you have stored all the CA certificates, or if you have received only one certificate from the CA, issue the SSLADMIN STORE command with the SERVER operand to store the server certificate:
     ```
     ssladmin store fn server
     ```
     The server certificate is given the label associated with the corresponding certificate request that already resides in the certificate database. If the certificate being stored is actually a certificate chain, the SSL server stores each CA certificate with a label based on the certificate owner's common name and organization and then stores the server certificate.

   For more information, see "SSLADMIN STORE Command" on page 544.

To see what CA certificates have been stored and under what names, issue the SSLADMIN QUERY command:

```
ssladmin query cert *
```

3. Restart the SSL server to load the updated certificate database into memory.

Creating a Self-Signed Certificate

For testing purposes, you can create a self-signed certificate that serves as both a server certificate and a CA certificate. Because the certificate is signed with the application server's private key instead of by a CA, it has no real security or authentication value.

To create a self-signed certificate:

1. Create a *label* X509INFO file containing information about the application server.

   See "X509INFO File" on page 531 for information about the structure of this file.
2. Issue the SSLADMIN SELF command to create the self-signed certificate:

   ```
   ssladmin self label keysize fm
   ```

   For more information, see "SSLADMIN SELF Command" on page 542.

   The self-signed certificate is automatically stored in the certificate database.
3. Send the self-signed certificate to the client.

   The client must have a copy of the self-signed certificate before it sends a connection request to initiate a test of the SSL environment. During the handshake, the SSL server sends the same self-signed certificate to the client as a server certificate. Because the self-signed certificate is not signed by a CA, the client cannot use a real CA certificate to verify it. Therefore, the client uses the previously-sent self-signed certificate as a CA certificate to verify the self-signed certificate sent during the handshake.

   Consult the documentation for the client application in use for information on how that client handles self-signed certificates.
4. Designate a secure port for testing purposes. For instructions, see "Step 4c: Designate the Secure Ports" on page 530. Make sure that the certificate label you specify on the PORT statement is the label of the self-signed certificate.
5. Restart the SSL server to load the updated certificate database into memory.

Now you are ready to receive a test connection request from the client.

Getting a Copy of a Certificate from the Database

Certificates and certificate requests cannot be examined once they are stored in the certificate database. To get a copy of a certificate or certificate request from the database and store it in a CMS file, issue one of the following SSLADMIN EXPORT commands:

```
ssladmin export fn fm certificate label
ssladmin export fn fm request label
```

For more information, see "SSLADMIN EXPORT Command" on page 536.

Deleting a Certificate or Request from the Database

If you want to remove a certificate or certificate request from the certificate database, use the SSLADMIN DELETE command. For more information, see "SSLADMIN DELETE Command" on page 535.

For example, suppose you created a self-signed certificate for testing, and now you want to send a certificate request to a CA for that same server. You probably want to use the same X509INFO input file. However, if you want to use the same label (the file name of the X509INFO file) for the certificate request, you must first delete the self-signed certificate from the database, because all labels in the database must be unique:

```
ssladmin delete certificate label
```

Remember to enter the label exactly as it is specified in the certificate database. If you are not sure how it is specified in the database, use the SSLADMIN QUERY command to get a list of all the labels:

```
ssladmin query certificate *
```

Note: The deleted certificate or certificate request remains active until you restart the SSL server to load the updated database into memory.

Rebuilding the Certificate Database

The objects in the certificate database are protected by a random internal password. You do not need to use or know this internal password to perform certificate database administration tasks. However, if you feel that the security of the database has been compromised, you can rebuild the database and force the generation of a new random password. Issue the SSLADMIN REGENERATE command:

```
ssladmin regenerate
```

SSL Server Administration

This section describes how to do various SSL server administration tasks, such as starting or stopping the server and turning tracing on or off.

Note: The SSL server must be running and TCPMAINT 591 must be accessed before you can issue SSLADMIN commands.

Starting the Server

The SSL server can be started in the same manner as other TCP/IP servers. For more information, see “Starting and Stopping TCP/IP Servers” on page 30.

Stopping the Server

To stop the SSL server, issue the SSLADMIN STOP command:

```
ssladmin stop
```

Using the Server Cache

It is possible to reuse the security parameters that were negotiated with a client during a previous secure session. Resuming a previously negotiated session cuts down on network traffic and CPU time and is therefore desirable. Resumption of a session can be initiated only by the client.

If the client wishes to resume a session, the client supplies the session ID to be resumed. If the SSL server finds this ID in the cache, it returns the same session ID to the client. Each side sends an encrypted message as a test. If this test is successful, no further negotiations are needed, and data flow begins.

If the SSL server cannot find the requested session ID in the cache, the entry may have expired (exceeded the CACHELIFE setting), or the entry was not put into the cache because the cache was full (exceeded the CACHESIZE setting). These settings are controlled by VMSSL operands.

You can check on the SSL server cache usage by issuing the SSLADMIN QUERY CACHE command. If the response from the command shows a large number of new sessions and a small number of resumed sessions, you might consider increasing the CACHELIFE or CACHESIZE setting, or both.

Tracing Server Activities

To begin tracing SSL server activities when the server starts, use the TRACE operand on the VMSSL command:

- You can specify TRACE as a VMSSL startup parameter in the DTCPARMS file. Tracing will begin each time the server is autologged until you modify the DTCPARMS file again to remove the TRACE startup parameter. See "Step 2: Update the DTCPARMS File" on page 527.
- You can specify TRACE on the VMSSL command when manually starting the server while logged on to the SSL server virtual machine. See "VMSSL Command" on page 527.

To dynamically start or stop tracing SSL server activities while the SSL server is running, use the SSLADMIN TRACE/NOTRACE command. See "SSLADMIN TRACE/NOTRACE Command" on page 545.

For information about trace output, see the *z/VM: TCP/IP Level 3A0 Diagnosis Guide*.

SSLADMIN DELETE Command

Use the SSLADMIN DELETE command to remove a certificate or certificate request from the certificate database.

```
►►──SSLadmin──DELete──┬─CERTificate─┬──label──────────────────────────►◄
                      └─REQuest─────┘
```

Operands

CERTificate
indicates that you are deleting a certificate.

REQuest
indicates that you are deleting a certificate request.

label
is the label associated with this certificate or certificate request in the certificate database.

Usage Notes

1. You must enter the label of the certificate or certificate request exactly as it is specified in the certificate database. Labels are case sensitive and may have imbedded blanks. Use the SSLADMIN QUERY command to determine how the label is specified in the certificate database.
2. The deleted certificate or certificate request remains active until you restart the SSL server to load the updated certificate database into memory.

SSLADMIN EXPORT Command

Use the SSLADMIN EXPORT command to retrieve a certificate or certificate request from the certificate database and store it in a CMS file.

```
►►──SSLadmin──EXPORT──fn──┬────┬──┬─CERTificate─┬──label──────────────►◄
                          └─fm─┘  └─REQuest─────┘
```

Operands

fn is the name to be used as the file name of the CMS file where the certificate or certificate request is stored:

- A certificate is stored in a file named *fn* X509CERT.
- A certificate request is stored in a file named *fn* CERTREQ.

fm is the file mode of the read/write disk or directory where the output file is to be written. If *fm* is not specified, the output file is written to the first read/write disk or directory in the CMS search order.

CERTificate
indicates that you are retrieving a certificate.

REQuest
indicates that you are retrieving a certificate request.

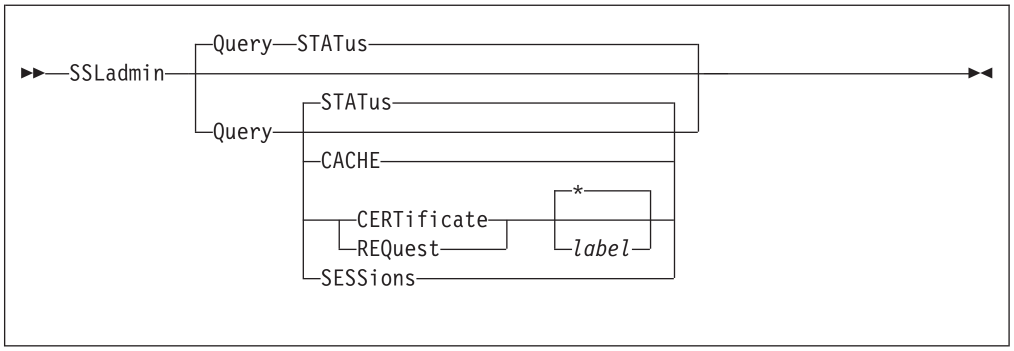
label
is the label associated with this certificate in the certificate database.

Usage Notes

1. If you are retrieving a certificate, the *fn* X509CERT file must not already exist on any accessed disk or directory.
2. If you are retrieving a certificate request, the *fn* CERTREQ file must not already exist on any accessed disk or directory.
3. The certificate or certificate request is stored in Base64-encoded format.
4. You must enter the label of the certificate or certificate request exactly as it is specified in the certificate database. Labels are case sensitive and may have imbedded blanks. Use the SSLADMIN QUERY command to determine how the label is specified in the certificate database.

SSLADMIN QUERY Command

Use the SSLADMIN QUERY command to display information about the SSL server, or about certificates or certificate requests in the certificate database.



Operands

STATus
: requests information about SSL server status. This is the default.

CACHE
: requests information about SSL server cache usage.

CERTificate
: requests information about certificates in the certificate database..

REQuest
: requests information about certificate requests in the certificate database.

label
: is the label associated with a specific certificate or certificate request you want information about.

  Specifying an asterisk (*) instead of a label retrieves a list of the labels of all the certificates or certificate requests contained in the certificate database. This is the default.

SESSions
: requests information about secure sessions.

Examples

SSLADMIN QUERY STATUS

When you request information about SSL server status, an information block is returned like this example:

```
Maximum number of sessions:  100
Number of active sessions:  4
Administration port:  9999
Cipher_suites included:  RC4_40_MD5 RC2_40_MD5 DES_40_SHA NULL
Cipher_suites exempted:  RC4_128_MD5 RC4_128_SHA DES_56_SHA RC4_56_SHA
Trace Settings:
  Normal: OFF
  Connections: ON
    Data: ON
  Flow: OFF
  Address: 255.255.255.255:0
  Connection: 0
```

The fields in this block supply the following information:

Maximum number of sessions
Maximum number of secure sessions the SSL server supports.

Number of active sessions
Current number of secure sessions.

Administration port
SSL administration port being used.

Cipher_suites included
Cipher suites the SSL server is allowed to use.

Cipher_suites exempted
Cipher suites the SSL server is not allowed to use.

Trace settings
Current SSL trace settings.

Note: The Data setting (shown in the example) is displayed only when the setting for Connections is ON.

SSLADMIN QUERY CACHE

When you request information about SSL server cache usage, an information block is returned like this example:

```
Cache entry life:  24
Total number of cache elements:  20
Number of new sessions:  3
Number of resumed sessions:  1
```

The fields in this block supply the following information:

Cache entry life
Number of hours that an entry can exist in the cache before it expires and can no longer be used.

Total number of cache entries
Maximum number of entries allowed in the cache.

Number of new sessions
Total number of secure sessions based on new handshake agreements between the SSL server and clients (that is, not resumed from cache entries) since the SSL server was started.

Number of resumed sessions
Total number of secure sessions resumed from cache entries since the SSL server was started.

Note: If this response shows a large number of new sessions and a small number or resumed sessions, you might consider increasing the CACHELIFE or CACHESIZE setting (VMSSL operands). For more information, see “Using the Server Cache” on page 534.

SSLADMIN QUERY CERTIFICATE *label*

When you request information about a specific certificate, an information block is returned like this example:

```
Certificate information:

           Label:    MYCERT
         Version:    3
   Serial number:    123458493939
       Issued by:
                     Class 1 Public Primary Certification Authority
                     VeriSign, Inc.
                     US
      Belongs to:
                     GDLVM7.ENDICOTT.IBM.COM
                     IBM Corporation
                     US
  Effective dates:   Aug 14 1999 to Aug 13 2000
            Type:    Server
        Key Size:    512
```

The fields in this block supply the following information:

Label Label associated with the certificate in the certificate database.

Version
X509 certificate version.

Serial number
Identifying number assigned to the certificate by the issuer.

Issued by
Information about the organization that signed the certificate.

Belongs to
Information about the entity that is validated by the certificate.

Effective dates
Period during which the certificate is valid.

Type Type of certificate: Server or CA.

Note: A self-signed certificate is identified as a server certificate.

Key Size
Number of bits in the public key.

SSLADMIN QUERY CERTIFICATE *

When you request a list of the labels of all the certificates in the certificate database, the list is returned like this example:

```
Labels are:

  1  – MYCERT
  2  – Thawte Personal Premium CA
  3  – Thawte Personal Freemail CA
  4  – Thawte Personal Basic CA
  5  – Thawte Premium Server CA
  6  – Thawte Server CA
  7  – Verisign Test CA Root Certificate
  8  – RSA Secure Server Certification Authority
  9  – Verisign Class 1 Public Primary Certification Authority
 10  – Verisign Class 2 Public Primary Certification Authority
 11  – Verisign Class 3 Public Primary Certification Authority

Disk usage 2%:  156 used  9044 available (1k-blocks)
```

The returned list includes the labels for server certificates, CA certificates, and self-signed certificates. The **Disk usage** line following the list indicates the current usage of the certificate database.

Note: Labels are case-sensitive and must be specified exactly as displayed, including imbedded blanks, in other SSLADMIN commands (such as SSLADMIN QUERY CERTIFICATE *label*).

SSLADMIN QUERY REQUEST *label*

When you request information about a specific certificate request, an information block is returned like this example:

```
Certificate request information:

       Label:     MYREQ
  Belongs to:
                  GDLVM7.ENDICOTT.IBM.COM
                  IBM Corporation
                  US
    Key Size:     512
```

The fields in this block supply the following information:

Label Label associated with the certificate request in the certificate database.

Belongs to
Information about the entity that is validated by the certificate.

Key Size
Number of bits in the public key.

SSLADMIN QUERY REQUEST *

When you request a list of the labels of all the certificate requests in the certificate database, the list is returned like this example:

```
Labels are:

  1  — MYREQ

Disk usage 2%:  156 used  9044 available (1k-blocks)
```

The **Disk usage** line following the list indicates the current usage of the certificate database.

Note: Labels are case-sensitive and must be specified exactly as displayed, including imbedded blanks, in other SSLADMIN commands (such as SSLADMIN QUERY REQUEST *label*).

SSLADMIN QUERY SESSIONS

When you request information about secure sessions, an information block is returned like this example:

```
Client_Socket_Address    Server_Socket_Address    Cipher      Label
---------------------    ---------------------    ------      -----
9.130.57.82:2505         9.130.249.45:423         RC4_40_MD5  MAJCERT2
9.130.57.82:2510         9.130.249.45:423         RC4_40_MD5  MAJCERT2
9.130.57.82:2516         9.130.249.45:423         RC4_40_MD5  MAJCERT2
9.130.57.82:2509         9.130.249.45:423         RC4_40_MD5  MAJCERT2
```

Each line in the information block represents a secure session. A secure session consists of two connections: the connection between the remote client and the SSL server, and the connection between the SSL server and the application server. The fields in this block supply the following information:

Client_Socket_Address
Address of the remote client in the secure session.

Server_Socket_Address
Address of the application server in the secure session.

Cipher
Cipher suite used for encrypted transmissions between the client and the SSL server.

Label Label of the certificate used to authenticate the application server.

Usage Notes

1. Information about the certificate database returned by this command does not reflect changes to the certificate database on disk that have not been loaded into memory. Restarting the SSL server loads the certificate database from disk into memory.

SSLADMIN REGENERATE Command

Use the SSLADMIN REGENERATE command to rebuild the certificate database. A new random internal password is generated and all the objects in the certificate database are password-encoded with this new password.

```
►►──SSLadmin──REGENerate──────────────────────────────────────────►◄
```

SSLADMIN REMOVE Command

Use the SSLADMIN REMOVE command to delete the certificate database. For example, you might want to refresh the database after testing. A new certificate database is immediately created, preloaded with some Well Known CA certificates.

```
►►──SSLadmin──REMOVE──────────────────────────────────────────────►◄
```

Usage Notes

1. Certificates and certificate requests in the old (removed) certificate database remain active until you restart the SSL server to load the new database into memory.

SSLADMIN REQUEST Command

Use the SSLADMIN REQUEST command to create a certificate request for an application server. As input to the command, you must provide information about the application server in a CMS file called *label* X509INFO. The file name of this file will be used as the label associated with the certificate request, and later the certificate itself, in the certificate database. For information about the contents of an X509INFO file, see “X509INFO File” on page 531.

A public/private key pair is created when the certificate request is generated. The request contains information, including the public key, that identifies the requestor, and the request is digitally signed with the private key.

The SSLADMIN REQUEST command stores the certificate request in the certificate database and saves a copy of the request in a CMS file called *label* CERTREQ.

```
►►──SSLadmin──REQuest──label──keysize──┬────┬──────────────────────────►◄
                                        └─fm─┘
```

Operands

label
: is the file name of the X509INFO input file. All accessed disks and directories are searched. This name will also be used as the file name of the CERTREQ output file.

keysize
: is the size, in bits, of the public and private keys that are created. The valid values are 512 or 1024.

fm
: is the file mode of the read/write disk or directory where the output file is to be written.

If *fm* is not specified:

- If the disk or directory containing the X509INFO input file is accessed read/write, the CERTREQ file is written to that disk.
- If the disk or directory containing the X509INFO input file is a read-only extension of a read/write disk or directory, the CERTREQ file is written to that read/write disk or directory.
- Otherwise, the CERTREQ file is written to the first read/write disk or directory in the CMS search order.

Usage Notes

1. Consult with the CA about what information they require in the certificate request and for instructions on how to send the certificate request.
2. The *label* CERTREQ file must not already exist on any accessed disk or directory, and *label* must not already exist as a certificate label in the certificate database.
3. The certificate request is stored in the CERTREQ file in Base64-encoded format.
4. Because *label* is a CMS file name, it is automatically uppercased, and the corresponding certificate request label is therefore uppercased in the certificate database.

SSLADMIN SELF Command

Use the SSLADMIN SELF command to create a self-signed certificate. Used primarily for testing purposes, a self-signed certificate is both a server certificate and a CA certificate. Because the certificate is signed with the server’s private key instead of by a CA, it has no real security or authentication value.

As input to the command, you must provide information about the application server in a CMS file called *label* X509INFO. The file name of this file will be used as the label associated with this certificate in the certificate database. For information about the contents of an X509INFO file, see "X509INFO File" on page 531.

The SSLADMIN SELF command stores the certificate in the certificate database and saves a copy of the certificate in a CMS file called *label* X509CERT.

```
►►──SSLadmin──SELF──label──keysize──┬────┬──────────────────────────►◄
                                    └─fm─┘
```

Operands

label
: is the file name of the X509INFO input file. All accessed disks and directories are searched. This name will also be used as the file name of the X509CERT output file.

keysize
: is the size, in bits, of the public and private keys that are created. The valid values are 512 or 1024.

fm
: is the file mode of the read/write disk on which the output file is to be written.

  If *fm* is not specified:
  - If the disk or directory containing the X509INFO input file is accessed read/write, the X509CERT file is written to that disk.
  - If the disk or directory containing the X509INFO input file is a read-only extension of a read/write disk or directory, the X509CERT file is written to that read/write disk or directory.
  - Otherwise, the X509CERT file is written to the first read/write disk or directory in the CMS search order.

Usage Notes

1. The *label* X509CERT file must not already exist on any accessed disk or directory, and *label* must not already exist as a certificate label in the certificate database.
2. The certificate is stored in the X509CERT file in Base64-encoded format.
3. Because *label* is a CMS file name, it is automatically uppercased, and the corresponding certificate label is therefore uppercased in the certificate database.
4. Clients that make use of SSL services generally have the certificates associated with the Well Known CAs in their certificate databases. The clients use those CA certificates to verify the server certificates sent during the handshake. When you create a self-signed certificate, you need to send that certificate to the client so the client can use it to verify the your server certificate. For more information, see "Creating a Self-Signed Certificate" on page 533.
5. You cannot use the self-signed certificate until the SSL server is restarted to load the updated certificate database into memory.

SSLADMIN STOP Command

Use the SSLADMIN STOP command to shut down the SSL server.

```
►►──SSLadmin──STOP───────────────────────────────────────────────────────►◄
```

Usage Notes

1. When the shutdown is complete, a disabled wait PSW is loaded. It is then safe to restart the server.

SSLADMIN STORE Command

Use the SSLADMIN STORE command to store a server certificate or CA certificate in the certificate database. You must first receive the certificate into a CMS file with a file type of X509CERT before you issue the SSLADMIN STORE command. The file name of the X509CERT file for a server certificate must match the corresponding certificate request in the certificate database.

```
                               ┌─SERver─────┐
►►──SSLadmin──STORE──fn────────┼────────────┼────────────────────────────►◄
                               └─CA──label──┘
```

Operands

fn is the file name of the X509CERT input file. All accessed disks and directories are searched.

SERver
indicates that you are storing a server certificate. This is the default.

CA
indicates that you are storing a CA certificate.

label
is the label to be associated with this CA certificate in the certificate database. This label can be any unique string up to 200 characters. It cannot begin or end with a blank, but it may contain imbedded blanks. It is case-sensitive.

Usage Notes

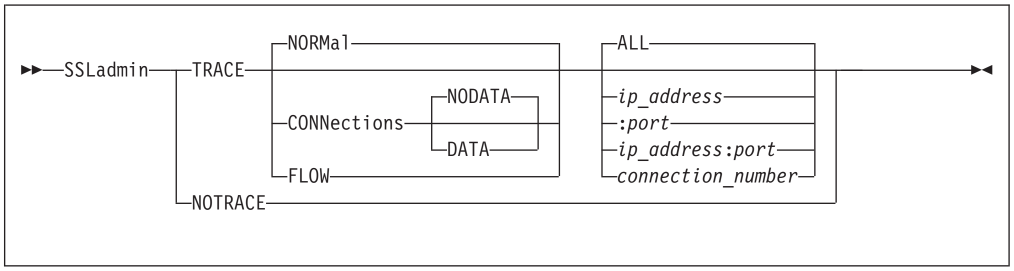
1. In response to a single certificate request, the CA may return one or more CA certificates along with the server certificate. Receive each certificate into a distinct X509CERT file. Because the CA certificates are used to verify the server certificate, you must issue the SSLADMIN STORE command with the CA operand to store each CA certificate with a unique label before you can issue the SSLADMIN STORE command with the SERVER operand to store the server certificate.
2. When you store a CA certificate, *label* must not already exist as a label in the certificate database.
3. If the CA returns only one certificate, it may be a server certificate or it may be a certificate chain. A certificate chain consists of a server certificate plus one or

more CA certificates. After receiving the certificate into the X509CERT file, issue the SSLADMIN STORE command with the SERVER operand. If the certificate is actually a certificate chain, each CA certificate is stored with a label based on the certificate owner's common name and organization.

4. When a server certificate is stored, it is given the label associated with the corresponding certificate request in the certificate database.
5. You cannot use a certificate you have stored until the SSL server is restarted to load the updated certificate database into memory.

SSLADMIN TRACE/NOTRACE Command

Use the SSLADMIN TRACE/NOTRACE command to start or stop tracing SSL server activities while the server is running.



Operands

TRACE
: specifies that tracing is to be performed.

NORMAL
: specifies that a trace entry is recorded to indicate a successful connection. This is the default if TRACE is specified.

CONNECTIONS
: specifies that a trace entry is recorded for connection state changes and handshake results.

NODATA
: specifies that no data is displayed for send and receive trace entries. This is the default if CONNECTIONS is specified.

DATA
: specifies that the first 20 bytes of data are displayed for send and receive trace entries.

FLOW
: specifies that flow of control and system activity are traced.

ALL
: specifies that tracing is done for all connections. This is the default if TRACE is specified.

ip_address
: specifies that tracing is done only for activity associated with this IP address.

:*port*
: specifies that tracing is done only for activity associated with this port.

connection_number
: specifies that tracing is done only for activity associated with this connection number.

NOTRACE
specifies that all tracing is turned off.

Usage Notes

1. For information about trace output, see the *z/VM: TCP/IP Level 3A0 Diagnosis Guide*.
2. This command turns tracing on or off for the current SSL server session only. If TRACE is specified as a VMSSL startup parameter in your DTCPARMS file, and you issue SSLADMIN NOTRACE to turn tracing off, tracing begins again each time the SSL server is restarted.

Chapter 24. Configuring the TFTP Server

The TFTP daemon (server) transfers files between the BFS (Byte File System) and TFTP clients. The TFTP server supports access to files maintained in a BFS directory structure that is mounted during initialization.

To configure the TFTP server, you must perform the following steps:

TFTP Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the TFTP server.
3. Review and address additional configuration considerations.
4. Create the TFTPD PERMLIST data file.
5. Create the TFTPD USERLIST data file.

Dynamic Operations

The TFTP server provides a console subcommand interface that allows you to perform various server administration tasks. For more information see "Dynamic Server Operation" on page 551.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
  TFTPD  0
```

The TFTP server requires port UDP 69 to be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
PORT
  69  UDP TFTPD  ; TFTPD Server
```

MTU Considerations

You should also verify that the Maximum Transmission Unit (MTU) size — specified on the GATEWAY statements for the subnetworks on which your IBM Network Stations will operate — is as large as can be reasonably be specified for your environment. Ideally, the MTU size should be greater than or equal to the block size used by TFTPD to load files to the IBM Network Station (by default, 8192 bytes), so that high performance is attained when files are transferred using TFTP.

However, you should use care when you specify or change the MTU size for any network or subnetwork. See the *z/VM: Performance* book prior to making changes of this nature.

Step 2: Update the DTCPARMS File

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

```
:Nick.TFTPD
  :Mount.
  :Parms.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see "Chapter 3. General TCP/IP Server Configuration" on page 17.

Note: You should modify the DTCPARMS file for the TFTPD server if you:
- Mount a BFS directory other than: **/../VMBFS:VMSYS:ROOT/**
- Change the parameters passed to the TFTPD command.

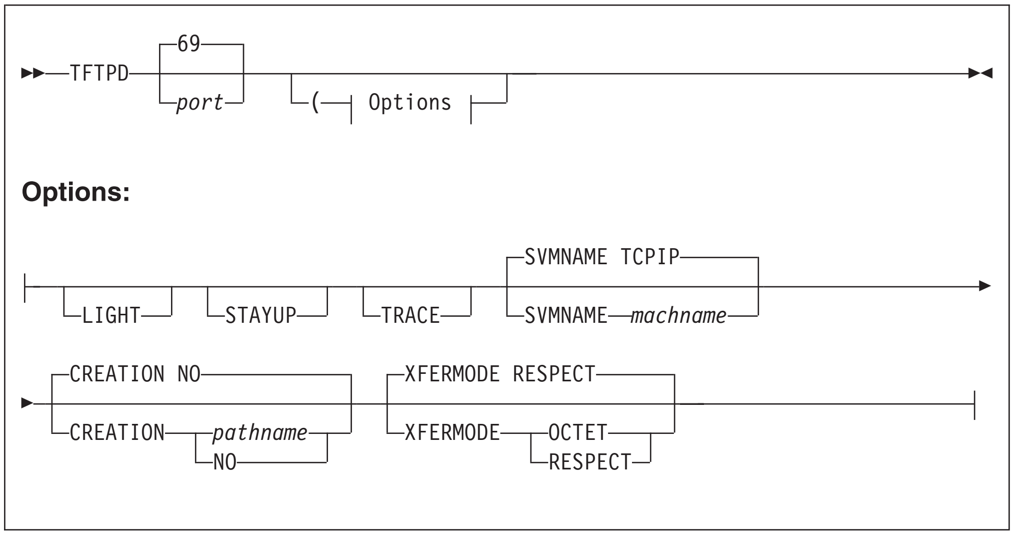
If no parameters are specified on the **:Parms.** tag in the DTCPARMS file, TFTPD is initialized with the following parameters:

```
69 ( SVMNAME TCPIP XFERMODE OCTET
```

TFTPD Command

The TFTP daemon (the TFTPD server) transfers files between the Byte File System (BFS) and TFTP clients. TFTPD supports access to files maintained in a Byte File System directory structure that is mounted during initialization. Files may be specified as absolute or relative path names; fully qualified path names are rejected.

TFTP services are initiated using the TFTPD command:



Specify TFTPD command operands as **:Parms**. tag startup parameters in your DTCPARMS file.

Operands

port
The port on which the TFTP daemon should listen for requests.

LIGHT
Indicates that the TFTP daemon should operate in light-traffic mode. See "LIGHT Subcommand" on page 557 for further information about this mode.

STAYUP
Indicates that the TFTP daemon should continue to operate across VM TCP/IP failures.

TRACE
Indicates that the TFTP daemon should display debug information as requests are processed.

SVMNAME *machname*
Controls the name of the TCP/IP server machine with which the TFTP daemon should interact.

CREATION NO
Indicates that clients may not create files in the Byte File System.

CREATION *pathname*
Indicates that clients may create files in the Byte File System on the specified directory or subdirectories built off of that directory; *pathname* may be an absolute path name or a relative path name of an existing directory in the Byte File System. See the "Usage Notes" section in "CREATION Subcommand" on page 555 for additional information about this function.

XFERMODE RESPECT
Indicates that the TFTP server should respect the transfer mode requested by the client. All data in the Byte File System is assumed to be EBCDIC data. When the client requests that the file be transferred as NETASCII, the data is converted between EBCDIC and NETASCII.

XFERMODE OCTET
Indicates that the TFTP server should ignore the transfer mode requested by the client. No translation will be performed on the transferred data. This allows data to be stored in the Byte File System in ASCII format.

Usage Notes

- The TFTP server used to provide TFTP support does not need to reside on the VM system that participates in the subnet in which this activity occurs. However, for performance reasons, IBM recommends this server to be present there, as this eliminates the need for router involvement in order to forward information between the client and the VM system where the TFTP server is located.
- The TFTP server reads files into memory (cache) before they are sent to a client. Similarly, it receives a complete file before that file is written to the Byte File System. In addition, the permanent file list (TFTPD PERMLIST) identifies files that should be kept in memory. Due to the fact that files are maintained in memory, you may need to increase the virtual machine size to accommodate the number and size of files that your machine handles. Errors due to insufficient storage in the TFTP server will cause read or write operations to fail.
- The **STAYUP** option is needed only when the TCP/IP machine does not contain an entry for the virtual machine running TFTPD.

Step 3: Additional TFTPD Configuration Considerations

Restricting TFTP Access to Files

If you intend to restrict TFTP access to certain files maintained in the Byte File System (BFS), or make use of the TFTPD **CREATION** function, you will need to ensure the TFTP server is configured as a POSIX "superuser". To allow this capability, the following statement must be included in the CP directory entry for the TFTP server virtual machine:

```
POSIXINFO UID 0 GID 0
```

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

Changing the TFTP Server BFS Directory Default

Because the TFTP server provides BFS-specific file support, a BFS directory must be mounted by this server during initialization so that the necessary OpenExtensions Shell and Utilities commands are available. By default, this directory is:

```
/../VMBFS:VMSYS:ROOT/
```

If the OpenExtensions Shell and Utilities has been installed in a non-default file pool (that is, different from VMSYS), you will need to change the default mount directory to reflect the file pool used for your root file space and root directory. This can be done through use of the **:Parms.** tag in the DTCPARMS file for the TFTP server.

Controlling TFTP Server Data Translation

The IBM Network Station support provided by APAR PQ02301 requires the TFTP server to not perform any EBCDIC to NETASCII translation of the data it transfers. Thus, the **XFERMODE OCTET** operand must be specified when the TFTP server is initialized; this is the default. You can change this default by specifying the desired transfer mode in the **:PARMS** tag of your DTCPARMS file.

Collecting TFTP Server Monitor Data

The TFTP server creates monitor data during its operation. To allow this data to be collected by CP, the APPLMON option must be included in the TFTPD directory entry. See "TFTP Server APPLDATA Monitor Records" on page 562 for more information about the monitor records created by TFTP server.

Step 4: Create the TFTPD PERMLIST Data File

The TFTPD PERMLIST file identifies files which the TFTP server should maintain in cache, after an initial access to one of these files by a client. The TFTPD PERMLIST file format is:

- One line per pathname.
- Blank lines and lines where the first word is an asterisk (*) are ignored by the TFTP server.

The TFTPD PERMLIST file should be created and maintained on the TCPMAINT 198 minidisk.

A sample TFTPD PERMLIST file follows:

```
* ---------------------------------------------------------------------
* Sample TFTPD PERMLIST file
* ---------------------------------------------------------------------
* This file identifies files which are to be maintained in cache by the
* TFTPD server after an initial access by a client.

  /Xncd
  /mods/login.nws
  /mods/show.nws
```

Notes:

1. Maintaining files in memory improves TFTP server performance at the expense of increased virtual storage utilization. You may need to increase virtual storage for the TFTP server to account for the number and size of the files listed in the TFTPD PERMLIST file.
2. Absolute path names should be used to identify files in the TFTPD PERMLIST file.
3. For more information about how to manage which files the TFTP server maintains in cache, see “LOADPERM Subcommand” on page 558 and “DROPFILE Subcommand” on page 556.

Step 5: Create the TFTPD USERLIST Data File

The TFTPD USERLIST file associates client IP addresses with a VM system user ID. This mapping allows an administrator to control user access to various files via the POSIX user ID (UID) and group ID (GID) values that correspond to the VM user IDs identified within this file. The TFTPD USERLIST file format is:

- One line per IP address and user ID pair. This line consists of two blank delimited words, where the:
  - first word is the client IP address, in dotted decimal notation.
  - second word is the VM user ID that should be associated with this IP address. A user ID may be associated with more than one client IP address.
- Blank lines and lines where the first word is an asterisk (*) are ignored by the TFTP server.

The TFTPD USERLIST file should be created and maintained on the TCPMAINT 198 minidisk.

A sample TFTPD USERLIST file follows:

```
* ---------------------------------------------------------------------
* Sample TFTPD USERLIST file
* ---------------------------------------------------------------------
* This file maps client IP addresses to VM system user IDs.
* IP addresses within this file must be specified using dotted decimal
* notation (for example, 123.45.67.89).

  9.130.21.12  GEDDYLEE
  9.130.21.33  ROCKETMAN
  9.130.21.61  DYLANBOB
  9.130.21.19  STELYDAN
```

Dynamic Server Operation

| Some configuration attributes (such as tracing and exit enabling) can be changed
| during server operation by using the TFTP subcommands described in the next
| section. In addition, certain server activities can be queried or changed by
| subcommands.

TFTPD Subcommands

The TFTPD subcommands are listed in Table 30. This table provides the shortest abbreviation, a description, and a page reference for more information for each TFTPD subcommand.

Table 30. TFTPD Subcommands

Subcommand	Minimum Abbreviation	Description	Page
CACHE	CACHE	Displays a list of cached files.	552
CLIENTS	CLIENTS	Displays a list of clients being served by the TFTP server.	554
CMS	CMS	Passes a command to CMS for execution.	555
CREATION	CREATION	Specifies a directory on which clients may create files.	555
DROPFILE	DROPFILE	Removes a specific file from cache.	556
EXIT	EXIT	Stops the TFTP server and its processing. EXIT is equivalent to QUIT and STOP.	557
HELP	HELP	Displays a summary of TFTPD subcommands.	557
LIGHT	LIGHT	Toggles LIGHT processing mode of the TFTP server.	557
LOADPERM	LOADPERM	Loads the TFTPD PERMLIST file, which lists files to be maintained in cache.	558
LOADUSER	LOADUSER	Loads the user mapping file, TFTPD USERLIST.	559
QUIT	QUIT	Stops the TFTP server and its processing. QUIT is equivalent to EXIT and STOP.	559
STAYUP	STAYUP	Toggles the STAYUP mode of the TFTP server.	560
STOP	STOP	Stops the TFTP server and its processing. STOP is equivalent to EXIT and QUIT.	560
TRACE	TRACE	Toggles the TRACE mode of the TFTP server.	560
XFERMODE	XFERMODE	Determines whether NETASCII file translation should be performed.	561

CACHE Subcommand

Use the CACHE subcommand to display the list of cached files.

```
►►──CACHE──────────────────────────────────────────────────────────►◄
```

Usage Notes

- Once a file is marked to be dropped from cache via the **DROPFILE** subcommand, the next new request by a client to read that file will cause a new version of the file to be obtained from the Byte File System.
- If a client's request for a file "can be satisfied" from memory, such is done. Otherwise, a copy of the file is loaded from DASD into memory and that copy is used.

  The phrase "can be satisfied" is true if:
  - the requested file is in memory, and
  - it is in the correct format (either NETASCII or OCTET), and
  - it is not marked to be dropped by the **DROPFILE** subcommand.

It is possible that more than one client may use the same memory image of a given file, even if that file is not in the permanent file list (the TFTPD PERMLIST file). This could happen if the timing is such that a file already present in memory to satisfy one client's request can also satisfy the new client's request.

- A file is dropped from cache when no clients are using it, unless it is marked as "permanent", by appearing in the permanent file list.
- To keep a file in memory even though no clients are using it, the file must be listed in the TFTPD PERMLIST file.

  The permanent file list is loaded when the TFTP daemon is initialized, or when the **LOADPERM** subcommand is issued.
- A file is read from DASD only when it is requested by a client. There is no provision to "pre-load" a file into cache before it is requested by a client.
- When at least one file is cached, a table of cached files is displayed. This table has the following format:

  ```
  PD  InProg  M  Filesize  Pathname
  pd  iiiii   m  filesize  pathname
  ```

  where:

  pd is a two digit field with a 1 or 0 possible for each position, indicating "yes" or "no", respectively.

  - The first digit, *p*, indicates whether the file is being maintained permanently in storage (marked as "permanent").
  - The second digit *d*, indicates whether the file will be dropped from cache once its current use count goes to zero. [3]

    If *d* is 1, the next new request by a client to read this file will cause the TFTP daemon to obtain a new copy from the Byte File System. If *d* is 0, this file will be dropped from cache as soon as its use count goes to zero.

  iiiii
  is the number of clients to which the file is being sent.

  m is the transfer mode, either **n** for NETASCII, or **o** for OCTET.

  filesize
  is the size of the file, in bytes.

  pathname
  is the name of the file.
- When no files are maintained in memory, the following message is displayed:

  ```
  No files cached.
  ```

3. This information is useful only if the value of *p* is 1, indicating that the file is in the permanent file list.

CLIENTS Subcommand

Use the CLIENTS subcommand to display the list of clients that are currently being served by TFTPD.

```
►►──CLIENTS──────────────────────────────────────────────────────────────►◄
```

Usage Notes

- When at least one client is being served, a table of such clients is displayed. This table has the following format:

  ```
  Client          Userid    Bytes     BlkSz  DM  Pathname
  ipaddr          userid    bytes     blksz  dm  pathname
  ```

  ipaddr
  is the IP address of the client being served; this address is shown in dotted decimal notation (for example, xxx.xxx.xxx.xxx).

  userid
  is the user ID associated with this client. If the client is not associated with any user ID, this field is blank.

  bytes
  is the number of bytes transferred thus far between this client and the TFTP daemon.

  blksz
  is the block size being used for data transfer to this client.

  dm
  is a two character field indicating the direction of file transfer and the transfer mode.

  The first character, *d*, is the direction of transfer. *d* will have the value **w** if the transfer is from the IBM Network Station to the Byte File System (a client write request); **r** is used to indicate a transfer from the Byte File System to the client (a client read request).

  The second character, *m*, is the transfer mode, either **n** for NETASCII, or **o** for OCTET.

  pathname
  is the name of the file being transferred.

- When no clients are being served, the following message is displayed:

  ```
  No clients found.
  ```

CMS Subcommand

Use the CMS subcommand to issue a command to CMS.

```
►►──┬─────┬──cms_command──────────────────────────────────────►◄
    └─CMS─┘
```

Operands

cms_command
: is the CMS command to be issued.

Usage Notes

- Do not issue any CMS command that would take considerable time to execute (for example, XEDIT). While the CMS command executes, the server does not respond to requests.
- The **CMS** keyword is usually not required because the daemon will pass any command string that is not recognized as a TFTPD subcommand to CMS. The **CMS** keyword is used to identify CMS commands that would normally be interpreted as a TFTPD subcommand, for example, **TRACE**.

CREATION Subcommand

Use the CREATION subcommand to specify a directory on which clients may create files.

```
►►──CREATION──┬─pathname─┬────────────────────────────────────►◄
              └─NO───────┘
```

Operands

pathname
: is the name of the directory on which files may be created. In addition, files may be created on any subdirectory of the specified directory; *pathname* may be an absolute path name or a relative path name of an existing directory in the byte file system.

NO
: disables the ability for clients to create files.

Usage Notes

- Fully-qualified path names are not permitted for *pathname*.
- If *pathname* is specified as a relative pathname (does not begin with a slash—"/"), the path name of the creation directory is constructed each time a write request is processed for a file that does not exist.

The OpenExtensions **OPENVM SET DIRECTORY** command allows you to change the current working directory. The current working directory is used when resolving a relative address into an absolute address.

- The TFTP server virtual machine must be a POSIX "superuser" in order to create files for clients.
- In addition to the request to create a file on the creation directory or a subdirectory, a client must have write permission for the target directory for the file on a client's behalf.
- Files created by TFTPD are assigned the POSIX user ID (UID) and group ID (GID) of the VM user ID that corresponds to the client, as defined in the TFTPD USERLIST file. If the client is not associated with a VM user ID, the created file is assigned a POSIX UID of -1 and a GID of -1.
- The permission assigned to a created file depends upon the POSIX UID and GID values associated with the client's corresponding VM user ID:
  - Owner permission is set according to the corresponding user ID UID value. If the UID is -1, no Owner permission bits are turned on; otherwise, Owner permission is set to read and write (rw-).
  - Group permission is set according to the corresponding user ID GID value. If the GID is -1, no Group permission bits are turned on; otherwise, Group permission is set to read and write (rw-).
  - Other permission is set according to the corresponding user ID UID value. If the UID is -1, Other permission is set to read and write; otherwise, Other permission is neither read, write, nor executable (---).

The OpenExtensions **OPENVM SET MASK** command may be used to further restrict the permissions that are set when a file is created; this command specifies permissions which are to be "screened out". For example, the **OPENVM SET MASK** command could be added to the TFTPD profile exit (TFTPDXIT EXEC) to screen out Group write permission, so that files created on behalf of a client that has a GID other than -1 are created with a permission of read (r--) instead of read and write (rw-). This would mean that only the owner of the file could write that file; other members of the owner's Group would only be able to read it.

DROPFILE Subcommand

Use the DROPFILE subcommand to remove a file from cache after the current usage is complete.

```
►►──DROPFILE──pathname──────────────────────────────────────────────►◄
```

Operands

pathname
is the absolute path name of the file that should be dropped.

Usage Notes

- If the specified file is currently in use and is not in the permanent file list, that file will be removed from cache after the current usage is complete.

If the specified file is currently in use and is in the permanent file list, that file is marked as having been dropped. The first new request for access to this file will cause a new copy to be obtained from the Byte File System.

- A file can be requested by a client in two different forms, either NETASCII or OCTET; both forms of the file can reside in cache at the same time. In such a case, only the first instance of the file encountered in the cache will be dropped by the DROPFILE subcommand.

EXIT Subcommand

Use the EXIT subcommand to stop the TFTP daemon. This subcommand is equivalent to the **QUIT** and **STOP** subcommands.

```
►►──EXIT──────────────────────────────────────────────────────────────►◄
```

Operands

The EXIT subcommand has no operands.

HELP Subcommand

Use the HELP subcommand to display a brief description of available subcommands.

```
►►──HELP──────────────────────────────────────────────────────────────►◄
```

Operands

The HELP subcommand has no operands.

LIGHT Subcommand

Use the LIGHT subcommand to toggle the light-traffic mode in the TFTP daemon.

```
►►──LIGHT─────────────────────────────────────────────────────────────►◄
```

Operands

The LIGHT subcommand has no operands.

Usage Notes

- In light-traffic mode, the daemon works on the assumption that it will process requests faster than they will queue up. In this mode, the server will receive one datagram and then return to wait for an indication that another datagram has arrived.

  In heavy-traffic mode, the daemon works on the assumption that most often, it will be kept busy by a queue of datagrams. In this mode, the server will continue attempts to receive additional datagrams until it gets a “no more datagrams” response (EWOULDBLOCK error value), and then waits for another datagram to arrive.
- The following is displayed upon completion of this subcommand:

  ```
  LIGHT is now t
  ```

  where *t* is 0 if LIGHT mode is off; 1 if LIGHT mode is on.

LOADPERM Subcommand

Use the LOADPERM subcommand to reload a copy of the permanent file list, TFTPD PERMLIST.

```
►►──LOADPERM──────────────────────────────────────────────────────────►◄
```

Operands

The LOADPERM subcommand has no operands.

Usage Notes

- Files listed in the permanent file list (TFTPD PERMLIST) are kept in memory after an initial access by a client. There is no provision to “pre-load” a file into cache before it is requested by a client.
- The TFTPD PERMLIST file format is:
  – One line per pathname.
  – Blank lines and lines where the first word is an asterisk (*) are ignored by TFTPD.
- Maintaining files in memory improves TFTP server performance at the expense of increased virtual storage utilization.
- The LOADPERM subcommand loads the first occurrence of the TFTPD PERMLIST file present in the search order.
- Absolute path names should be used to identify files in the TFTPD PERMLIST file.

LOADUSER Subcommand

Use the LOADUSER subcommand to load a copy of the user mapping file, TFTPD USERLIST.

```
►►──LOADUSER─────────────────────────────────────────────────────────────►◄
```

Operands

The LOADUSER subcommand has no operands.

Usage Notes

- The user mapping file (TFTPD USERLIST) associates client IP addresses with a VM system user ID. This mapping allows an administrator to control user access to various files via the POSIX user ID (UID) and group ID (GID) permissions that correspond to the VM user IDs identified within this file.
- The TFTPD USERLIST file format is:
  – One line per IP address and user ID pair. This line consists of two blank delimited words, where the:
    - first word is the client IP address, in dotted decimal notation.
    - second word is the VM user ID that should be associated with this IP address. A user ID may be associated with more than one client IP address.

  – Blank lines and lines where the first word is an (*) are ignored by TFTPD.
- The LOADUSER subcommand loads the first occurrence of the TFTPD USERLIST file present in the search order.

QUIT Subcommand

Use the QUIT subcommand to stop the TFTP daemon. This subcommand is equivalent to the **EXIT** and **STOP** subcommands.

```
►►──QUIT─────────────────────────────────────────────────────────────────►◄
```

Operands

The QUIT subcommand has no operands.

STAYUP Subcommand

Use the STAYUP subcommand to toggle the STAYUP mode in the TFTP daemon.

```
►►──STAYUP──────────────────────────────────────────────────────────►◄
```

If the daemon is already operating in STAYUP mode, it will cease operating in this mode and will end processing if a subsequent TCP/IP failure occurs. If the daemon is not operating in STAYUP mode, it will begin to ensure processing will not end if a subsequent TCP/IP failure occurs.

Operands

The STAYUP subcommand has no operands.

Usage Notes

- The following is displayed upon completion of the subcommand:
  ```
  STAYUP is now t
  ```
  where *s* is 0 if STAYUP mode is off; 1 if STAYUP mode is on.

STOP Subcommand

Use the STOP subcommand to stop the TFTP daemon. This subcommand is equivalent to the **EXIT** and **QUIT** subcommands.

```
►►──STOP────────────────────────────────────────────────────────────►◄
```

Operands

The STOP subcommand has no operands.

TRACE Subcommand

Use the TRACE subcommand to toggle the TRACE mode in the TFTP daemon. If the daemon is already operating in TRACE mode, it will cease displaying debug information as it processes requests. If the daemon is not operating in TRACE mode, it will display debug information as it processes requests.

```
►►──TRACE───────────────────────────────────────────────────────────►◄
```

Operands

The TRACE subcommand has no operands.

Usage Notes

- The following is displayed upon completion of this subcommand:

  ```
  TRACE is now t
  ```

  where *t* is 0 if TRACE mode is off; 1 if TRACE mode is on.
- See the *TCP/IP Diagnosis Guide* for a description of TFTP server trace output.

XFERMODE Subcommand

Use the XFERMODE subcommand to indicate whether the TFTP daemon should translate data that is sent or received with a transfer mode of NETASCII.

```
►►──XFERMODE──┬─OCTET───┬─────────────────────────────────────────►◄
              └─RESPECT─┘
```

Operands

OCTET
: indicates that TFTPD should ignore the transfer mode requested by the client. No translation will be performed on the transferred data. This allows data to be stored in the Byte File System in ASCII format.

RESPECT
: indicates that TFTPD should respect the transfer mode requested by the client. All data in the Byte File System is assumed to be EBCDIC data. When the client requests that the file be transferred as NETASCII, the data is converted between EBCDIC and NETASCII.

TFTPD File Access Control

File access for clients served by the TFTP daemon is controlled by the Byte File System (BFS). A requester can only access files in the BFS subdirectory tree of the BFS directory mounted by the TFTP server at initialization. Additionally, the VM user ID that corresponds to a client must have the proper permission, so that the client can access these files.

For BFS files, there are three classes of users whose access can be controlled:

- Owner (the owner of a file or directory, whose UID matches the UID for the file)
- Group (a member of any group whose GID matches the GID for the file)
- Other (anyone else)

File access for these groups is controlled via a set of *permission* bits.

To perform a read or write task, a user ID have been granted the proper permission in at least one of these classes.

Usually, TFTP clients requesting access to Byte File System files are handled as “Other”, or public, requestors. However, it is possible to associate clients with a VM user ID so that additional access protection can be provided by the TFTP daemon. Once a client is associated with a specific user ID, that user ID's Owner (UID) and Group (GID) access is used to determine whether that client may access a file.

The TFTPD USERLIST file is used to associate, or map, client IP addresses to a VM user ID. This file is loaded when the TFTP server is initialized, and can be reloaded by the **LOADUSER** subcommand. See "Step 5: Create the TFTPD USERLIST Data File" on page 551 and "LOADUSER Subcommand" on page 559 for more information about the TFTPD USERLIST file and how to reload it.

TFTP Server APPLDATA Monitor Records

PI

The TFTP server contributes to the CP monitor data by using the APPLDATA call class. However, to allow this, the APPLMON directory option must be specified in the CP directory entry for the TFTP server virtual machine.

To begin data collection, use the CP MONITOR command to enable the APPLDATA domain, and to start event monitoring. See *z/VM: CP Command and Utility Reference* and *z/VM: Performance* for more information about the CP MONITOR command and performance monitoring.

The data records for servers are domain X'A' APPLDATA records, the general format of which is described in the *z/VM Monitor Records* file. Note that this file is located on your system's CP object disk (194) in a file named MONITOR LIST1403. Each data record is preceded by a record header, which is described in Table 31.

Every data record consists of these parts:
- CP header data
- TFTPD application data

The CP header data consists of the following:

Table 31. CP Header Data

Data Item	**Number of Bytes**
Byte offset to application data relative to start of this record	2
Length in bytes of application data	2
User ID of the server machine (in EBCDIC)	8
Product identification (in EBCDIC). For TFTP server records, this field contains "5735FALTFT010100".	16
Status	1
Reserved	3

Following the CP header data is the counter data.

Table 32 on page 563 shows record layout for the TFTPD-supplied application data. The offset values listed are the offsets into the application data area of the monitor record (field APLSDT_ADATA). Always use the byte offset and length fields in the standard domain 10 records to locate the start and end of the application data within the record.

Table 32. TFTPD APPLDATA Data

Offset					
Dec	Hex	Type	Len	Name	Description
0	(0)	SIGNED	4	mon_rrq_count	The total number of read requests, or RRQs (downloads).
4	(4)	CHARACTER	8	mon_rrq_time	The total elapsed time spent processing read requests. Accumulated in TOD clock units.
12	(C)	UNSIGNED	8	mon_rrq_bytes	The total number of bytes sent by the server.
20	(14)	SIGNED	4	mon_rrq_pkts_in	The total number of UDP packets received in association with read requests.
24	(18)	SIGNED	4	mon_rrq_pkts_out	The total number of UDP packets sent in association with read requests.
28	(1C)	SIGNED	4	mon_rrq_misses	The total number of times a read request could not be satisfied from cache.
32	(20)	CHARACTER	32	*	Reserved and available for IBM use.
64	(40)	SIGNED	4	mon_wrq_counts	The total number of write requests, or WRQs (uploads).
68	(44)	CHARACTER	8	mon_wrq_time	The total elapsed time spent processing write requests. Accumulated in TOD clock units.
76	(4C)	UNSIGNED	8	mon_wrq_bytes	The total number of bytes received by the server.
84	(54)	SIGNED	4	mon_wrq_pkts_in	The total number of UDP packets received in association with write requests.
88	(58)	SIGNED	4	mon_wrq_pkts_out	The total number of UDP packets sent in association with write requests.
92	(5C)	SIGNED	4	mon_wrq_misses	The total number of times a write request could not be satisfied from cache. This will always be zero.
96	(60)	CHARACTER	32	*	Reserved and available for IBM use.
128	(80)	SIGNED	4	mon_abn_count	The total number of transactions that did not complete or that abnormally ended.
132	(84)	UNSIGNED	8	mon_abn_bytes	The total number of bytes sent with transactions that failed.
140	(8C)	UNSIGNED	4	mon_abn_pkts_in	The total number of UDP packets received in association with failed transactions.
144	(90)	SIGNED	4	mon_abn_pkts_out	The total number of UDP packets sent in association with failed transactions.

Table 32. TFTPD APPLDATA Data (continued)

Offset					
Dec	Hex	Type	Len	Name	Description
148	(94)	CHARACTER	44	*	Reserved and available for IBM use.
192	(C0)	SIGNED	4	mon_timeouts	The total number of times the TFTP server timed out while waiting for an acknowledgement.
196	(C4)	CHARACTER	8	*	Reserved for future IBM use.

PI end

Chapter 25. Configuring the UFT Server

This chapter describes how to configure the Universal File Transfer (UFT) server (daemon). To configure the UFT server virtual machine, you must perform the following steps:

UFT Server Configuration Steps

1. Update the TCPIP server configuration file.
2. Update the DTCPARMS file for the UFT server.
3. Update the TCPIP DATA file.
4. Customize the UFTD CONFIG file
5. Perform advanced UFT server configuration, if needed.

Dynamic System Operation

The UFT server provides a console subcommand interface that allows you to perform various server administration tasks. For more information see “Dynamic Server Operation” on page 572.

Step 1: Update PROFILE TCPIP

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

```
AUTOLOG
   UFTD    0
```

The UFT server requires port TCP 608 to be reserved for it. Verify that the following statement has been added to your TCPIP server configuration file as well:

```
  PORT
    608  TCP UFTD    ; UFTD Server
```

Step 2: Update the DTCPARMS File

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

```
:nick.UFTD
  :Parms.
```

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

For more information about the DTCPARMS file, customizing servers, and server profile exits, see “Chapter 3. General TCP/IP Server Configuration” on page 17.

UFTD Command

UFT services are initiated using the UFTD command:

```
►►──UFTD──────────────────────────────────────────────────────────────►◄
```

The UFTD command has no operands.

Step 3: Update the TCPIP DATA File

To allow CMS users to be aware of the true origin of files received by the UFT server, add the following statement to the TCPIP DATA file:

```
UFTSERVERID UFTD
```

Step 4: Customize the UFTD CONFIG File

The UFT server uses one configuration file, UFTD CONFIG. This file is used to specify operational parameters for the UFT server virtual machine, such as:
- DBCS conversion options
- Identification of clients
- Maximum size of files received
- Name resolution of connecting clients
- Port that UFT listens on
- Protocol command user exits.

See “UFT Configuration File Statements” for detailed information about how to specify entries within this file. A sample UFT configuration file is provided as UFTD SCONFIG on the TCPMAINT 591 disk. Your customized UFT configuration file should be copied to the TCPMAINT 198 minidisk as UFTD CONFIG.

UFT Configuration File Statements

Specify UFT server parameters in the UFT configuration file as described in this section. Keep in mind the following when configuration statements are specified:
- Tokens are delimited by blanks and record boundaries.
- All characters to the right of, and including, a semicolon are treated as comments.

Note: GLOBALV values used by the UFT server are defined in the GLOBALV group **UFTD**. GLOBALV values override coded defaults, and are overridden by configuration file statements.

IDENTIFY Statement

The IDENTIFY statement specifies that the UFT server should attempt to identify the client by the ″identify protocol″ described by RFC 1413. If an IDENT server is available at the remote host the client is connecting from, the identity of that client (that is, a user ID) is obtained and saved.

```
    ┌─IDENTIFY YES─┐
►►──┤              ├──────────────────────────────────────────────►◄
    └─IDENTIFY NO──┘
```

Operands

YES Indicates that an attempt should be made to connect to the remote host's IDENT server and obtain the identity of the client. If the remote site has an IDENT server, the returned information is saved in the GLOBALV CLIENT variable. If TRACE ALL or TRACE IDENTIFY is active, the results are added to the trace output as well. If no IDENTIFY statement is specified in the configuration file, IDENTIFY YES is assumed.

NO Indicates that no request should be sent to the remote host's IDENT server.

MAXFILEBYTES Statement

The MAXFILEBYTES statement specifies the maximum size, in bytes, of mail that is accepted over a TCP connection.

```
    ┌─MAXFILEBYTES *─────┐
►►──┤                    ├────────────────────────────────────────►◄
    └─MAXFILEBYTES bytes─┘
```

Operands

* The default byte size limit. This has the special meaning of no limit.

bytes The maximum number of bytes to accept. Files arriving that are larger than this size are rejected. The limits for this statement are 1 and 2,147,483,647.

NSLOOKUP Statement

The NSLOOKUP statement specifies that resolution of the client connection address should be done. Once resolved, the results are saved in global variables and traced if TRACE ALL or TRACE NSLOOKUP is on.

```
    ┌─NSLOOKUP YES───────────────────────────┐
►►──┼────────────────────────────────────────┼────────────────────►◄
    ├─NSLOOKUP NO────────────────────────────┤
    │                               ┌─ON──┐  │
    └─NSLOOKUP EXIT──exit_name──────┤     ├──┘
                                    └─OFF─┘
```

Operands

YES Indicates that a DNS lookup should be performed against the client IP address. The resulting host name and IP address are saved in the GLOBALV variables HOSTNAME and HOSTADDR. If TRACE ALL or TRACE NSLOOKUP is active, the results are added to the trace output as well. If NSLOOKUP is not specified in the UFT configuration file, NSLOOKUP YES is assumed.

NO
Indicates that no DNS lookup of the client IP address should be performed.

EXIT *exit_name*
Indicates that a system administrator-provided exit exec should be called to determine what client IP address verification should be done. Based on the return code from the exec, the connection can be verified, not verified, or rejected. See “DNS Lookup Exit” on page 571 for details on the interface to this exit. The exec name defaults to **UFTNSLKX**.

ON
Indicates the exit exec should be invoked for all client resolution activity. This is the default.

OFF
Indicates the exit function should be initialized, but not yet invoked, for client resolution. A NSLOOKUP EXIT ON subcommand is required to initiate exit processing of client resolution.

PORT Statement

The PORT statement causes the UFT server to listen on a specific port. By convention, port number 608 is normally reserved for the UFT server to allow incoming mail requests to be accepted.

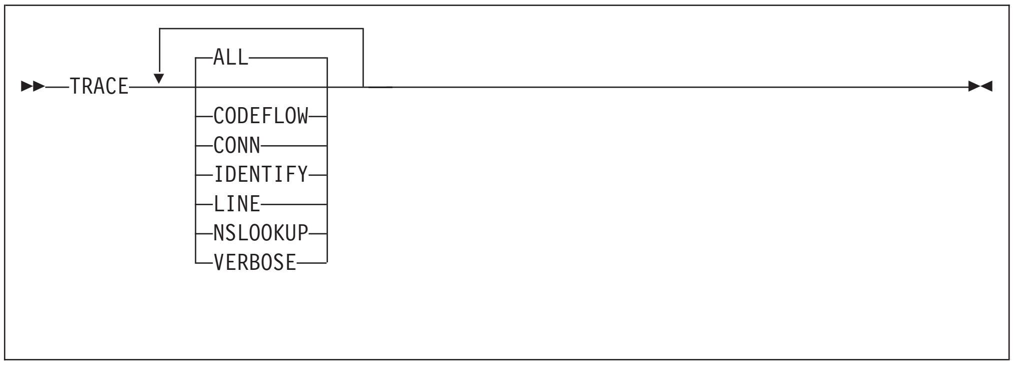
```
   ┌─PORT 608──────────┐
►►─┼───────────────────┼──────────────────────────────────────────────────►◄
   └─PORT port_number──┘
```

Operands

port_number
An integer in the range of 1 through 65,535 that specifies the port number to which the UFT server listens. The default is port 608.

TRACE Statement

The TRACE statement specifies which type of tracing should be turned on during initialization. Trace records are placed in the UFT DEBUG file.



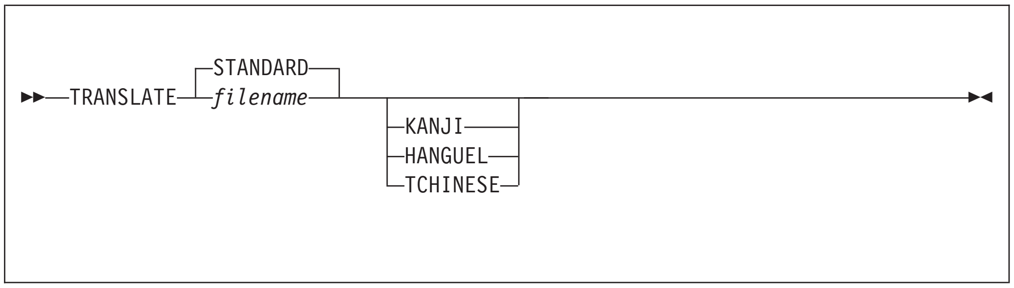
Operands

ALL	Initiates tracing of all types.
CODEFLOW	Initiates tracing of UFT code flow.
CONN	Initiates tracing of connectivity activity.
LINE	Initiates tracing of all commands and replies and their associated connection number.
IDENTIFY	Initiates tracing of the IDENTIFY function and results.
NSLOOKUP	Initiates tracing of the resolver function and results.
VERBOSE	Initiates tracing of a collection of UFT data, equivalent to specifying CODEFLOW, CONN and LINE.

If no trace statement is found in the UFT configuration file, no tracing is initiated. Multiple TRACE statements may be used to start more than one trace activity.

TRANSLATE Statement

The TRANSLATE statement specifies the file name of a SBCS or DBCS translation table to be used by the UFT server. When a translation file is specified, received ASCII file data is converted using the requested table defined within that file.



Operands

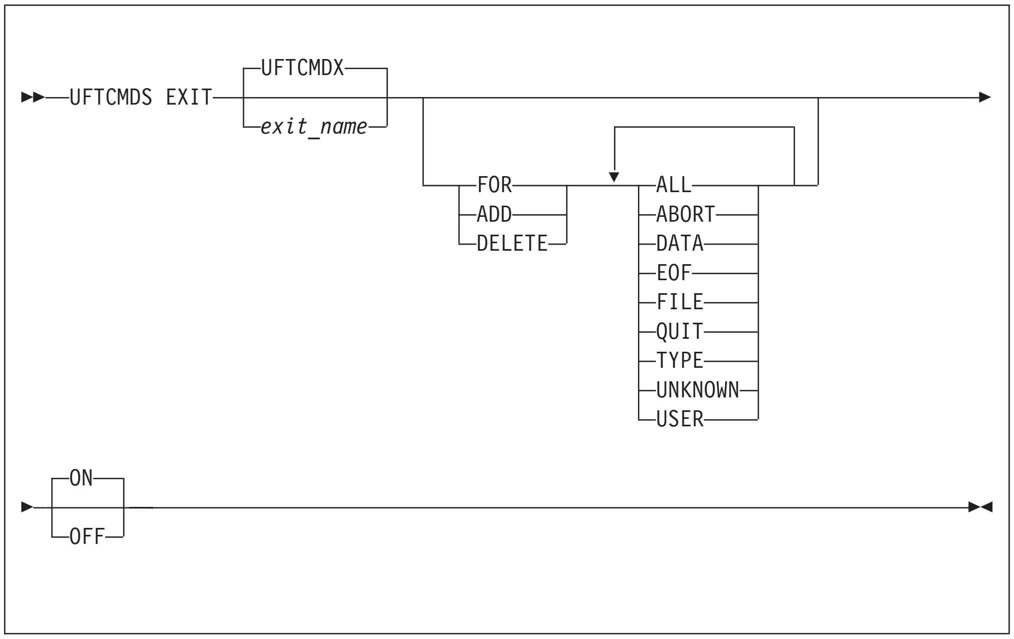
filename	Specifies the file name of the translation table to be used. The default file name is STANDARD and the file type defaults to TCPXLBIN unless a DBCS option (KANJI, HANGEUL or TCHINESE) is specified.
KANJI	Specifies the data received will contain DBCS strings and that a Kanji DBCS translation table should be used. The file type of the binary translation file must be TCPKJBIN.
HANGUEL	Specifies the data received will contain DBCS strings and that the

Hangeul DBCS translation table should be used. The file type of the binary translation file must be TCPHGBIN.

TCHINESE
Specifies the data received will contain DBCS strings and that the Traditional Chinese DBCS translation table should be used. The file type of the binary translation file must be TCPCHBIN.

UFTCMDS EXIT Statement

The UFTCMDS EXIT statement specifies which UFT protocol commands should be processed by an administrator-provided exit exec.



Operands

exit_name
Name of the exec to be called to handle any initiated protocol command exits. This exec is called on receipt of any commands for which the exit is enabled. See “Protocol Commands Exit” on page 571 for details on the exit interface. The default exec name is **UFTCMDX**.

FOR
Defines which UFT protocol commands are enabled for the exit. If the UFTCMDS exit was previously defined, any commands specified with FOR replace all previously enabled commands.

ADD
Defines which UFT protocol commands are to be added to any existing commands. The enabled commands will be the sum of any previously enabled commands and the commands specified with ADD.

DELETE
Defines which UFT protocol commands are to be deleted from exit processing. The exit remains active for any remaining enabled commands, if any.

command name
The *command name*, which consists of ALL, ABORT, DATA, EOF, FILE, QUIT, TYPE, UNKNOWN, and USER, identifies which protocol commands cause the exit to be called. For any individual command, any valid representation of the command or its synonyms, causes the exit to be called. Two keywords are not commands, but have special meaning. ALL indicates the exec is to

be called for all UFT commands eligible for exit processing, and UNKNOWN indicates the exit is to be called for any unknown commands received.

ON
Indicates the exit exec should be invoked for any enabled UFT commands. This is the default.

OFF
Indicates the exit function should be initialized, but not yet invoked, for any UFT protocol commands. A UFTCMDS EXIT ON command is then required to initiate exit processing of UFT protocol commands.

Step 5: Advanced Configuration Considerations

| The server exits described in this section can provide greater control over how files
| are processed by the UFT server. These exits might be used to:

- | reject files from a particular host, based on an IP address.
- | reject particular UFT protocol commands or control file processing performed by
  | the UFT server, based on available ″state″ information.

DNS Lookup Exit

The DNS lookup exit, when enabled, is called for each new UFT client connection made to the server.

DNS Lookup Exit Input

The following blank delimited arguments are passed to the exit:

Argument	Value	Format
1	Version of plist (1 currently)	Integer
2	Port numbers of UFT Server	Integer
3	IP Address of UFT Server	Character
4	Port number of the Client or 0 if not available	Integer
5	Source IP address of Client	Character
6	User value 1 (null)	Any

Return Codes

The UFT server recognizes one of the following exit return codes:

Return Code	Meaning
0	Accept client as valid (no verification)
1	Perform verification (same as NSLOOKUP ON)
2	Close connection
3	Disable the exit function

| A sample DNS lookup exit is provided as UFTNSLKX SEXEC on the TCPMAINT
| 198 minidisk. Your customized exit should be maintained on this same minidisk,
| with a file type of EXEC. For more information about the supplied DNS lookup
| exit, refer to the content of the supplied **UFTNSLKX SEXEC** file.

Protocol Commands Exit

The protocol commands exit, when enabled for a given command, is called once for each UFT protocol command received by the server.

Protocol Commands Exit Input

The following blank delimited arguments are passed to the exit:

Argument	Value	Format
1	Version of plist (1 currently)	Integer
2	Port number of UFTD Server	Integer
3	IP address of the UFTD Server	Character
4	Port number of the client or 0 if not available	Integer
5	Source IP address of client	Character
6	Client host domain name, (if resolved) or keyword ″UNKNOWN″	Character
7	Command being processed	Character
8	Length of command	Integer
9	Remainder of command buffer enclosed in single quotes	Character
10	Length of buffer	Integer
11	User value 1	Any
12	User value 2	Any

Note: User values 1 and 2 are initially null for the first call to the exit. Returned values are saved and passed as input to subsequent calls to the exit for the life of the connection. Once the connection is closed, the user values are reset to null for the next call on a new client connection.

Return Codes

The UFT server recognizes the following exit return codes:

Return Code	Meaning
0	Continue processing command
1	Reject this command as out of sequence (503)
2	Close the connection, abort transfer (526)
3	Return the response code contained in User value 1 to the client. The response code must exist in the UFT message repository (UFTUME) as a protocol response message.
4	Disable the exit function

| A sample protocol commands exit is provided as UFTCMDX SEXEC on the TCPMAINT 198 minidisk. Your customized exit should be maintained on this same minidisk, with a file type of EXEC. For more information about the supplied protocol commands exit, refer to the content of the supplied **UFTCMDX SEXEC** file.

Dynamic Server Operation

Some configuration attributes (such as tracing and exit enabling) can be changed during server operation by using the UFTD subcommands described in the next section. In addition, certain server activities can be queried or changed by subcommands. Any subcommand not understood by the UFT server is assumed to be a valid CMS command and is passed to the CMS command line for execution.

Issue UFTD subcommands at the UFT server console.

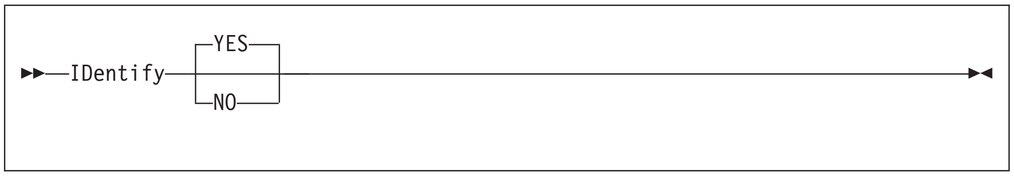
UFTD Subcommands

The UFTD subcommands are listed in Table 33. This table provides the shortest abbreviation, a description, and a page reference for more information for each UFTD subcommand.

Table 33. UFTD Subcommands

Subcommand	Minimum Abbreviation	Description	Page
IDENTIFY	ID	Causes the server to obtain the identity of a client.	573
NSLOOKUP	NSL	Causes the server to perform a DNS lookup.	573
QUERY	Q	Displays configuration, exit and tracing status.	574
QUIT	QUIT	Terminates the server.	574
STOP	STOP	Terminates the server.	574
TRACE	TR	Activates or deactivates server tracing.	575
UFTCMDS EXIT	UFTCMDS EXIT	Adds or removes protocol command exit points on the running system.	576

IDENTIFY Subcommand

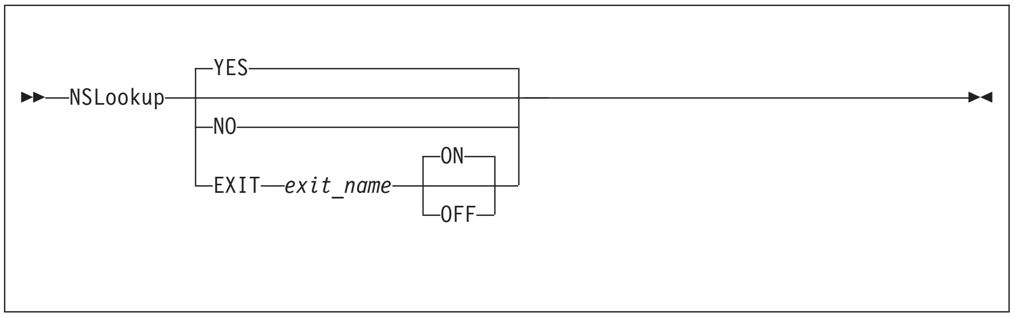


Operands

YES
Indicates that an attempt should be made to connect to the remote host's IDENT server and obtain the identity of the client. If the remote site has an IDENT server, the returned information is saved in the GLOBALV CLIENT variable. If TRACE ALL or TRACE IDENTIFY is active, the results are added to the trace output as well. This is the default.

NO
Indicates that no request should be sent to the remote host's IDENT server.

NSLOOKUP Subcommand



Operands

YES Indicates that a DNS lookup should be performed against the client IP address. The resulting host name and IP address are saved in the GLOBALV variables HOSTNAME and HOSTADDR. If TRACE ALL or TRACE NSLOOKUP is active, the results are added to the trace output as well. This is the default.

NO Indicates that no DNS lookup of the client IP address should be performed.

EXIT*exit _name*
Indicates that a system administrator-provided exit exec should be called to determine what client IP address verification should be done. Based on the return code from the exec, the connection can be verified, not verified, or rejected. See “DNS Lookup Exit” on page 571 for details on the interface to this exit. The exec name defaults to **UFTNSLKX**.

ON Indicates to start exit invocation for client resolution activity. This is the default.

OFF Indicates to stop exit invocation for client resolution activity.

The NSLOOKUP command always resets any existing settings. For example, issuing NSLOOKUP YES after NSLOOKUP EXIT has the effect of turning off the exit and turning on UFT client verification.

QUERY Subcommand

Use the QUERY subcommand to display configuration, user exit and tracing status.

```
            ┌─All───┐
►►──Query───┤       ├─────────────────────────────────────────────►◄
            ├─Exits─┤
            └─Trace─┘
```

Operands

All Displays all available configuration, server exit and tracing status.

Exits Displays status on all server exits, whether active or not, and the name of the exit routines in use (if applicable).

Trace Displays status for all trace modes.

QUIT Subcommand

Use the QUIT subcommand (or its synonym STOP) to terminate the server.

```
►►──QUIT──────────────────────────────────────────────────────────►◄
```

STOP Subcommand

Use the STOP subcommand (or its synonym QUIT) to terminate the server.

```
►►──STOP──────────────────────────────────────────────────────────────►◄
```

TRACE Subcommand

Use the TRACE subcommand to activate of deactivate tracing within the UFT server. Trace records are placed in the UFT DEBUG file.

```
                ┌─────────────────────┐
                │  ┌─ALL───────────┐  │
►►──TRace───────▼──┼───────────────┼──┴─────────────────────────────►◄
                   ├─END───────────┤
                   ├─CODEflow──────┤
                   ├─NOCODEflow────┤
                   ├─CONNectivity──┤
                   ├─NOCONNectivity┤
                   ├─IDentify──────┤
                   ├─NOIDentify────┤
                   ├─LIne──────────┤
                   ├─NOLIne────────┤
                   ├─NSlookup──────┤
                   ├─NONSlookup────┤
                   ├─VErbose───────┤
                   └─NOVErbose─────┘
```

Note: If TRACE statements are found in the UFT configuration file, the active tracing is the sum of those traces and any traces activated using the TRACE subcommand.

Operands

ALL	Initiates tracing of all types.
END	Terminates tracing of all types.
CODEflow	Initiates tracing of UFT code flow.
NOCODEflow	Terminates tracing of UFT code flow.
CONNectivity	Initiates tracing of connectivity activity.
NOCONNectivity	Terminates tracing of connectivity activity.
IDENTify	Initiates tracing of the IDENTIFY function and results.
NOIDENTify	Terminates tracing of the IDENTIFY function and results.
LINE	Initiates tracing of all commands and replies and their associated connection number.
NOLINE	Terminates tracing of all commands and replies and their associated connection number.
NSlookup	Initiates tracing of the resolver function and results.

NONSlookup Terminates tracing of the resolver function and results.

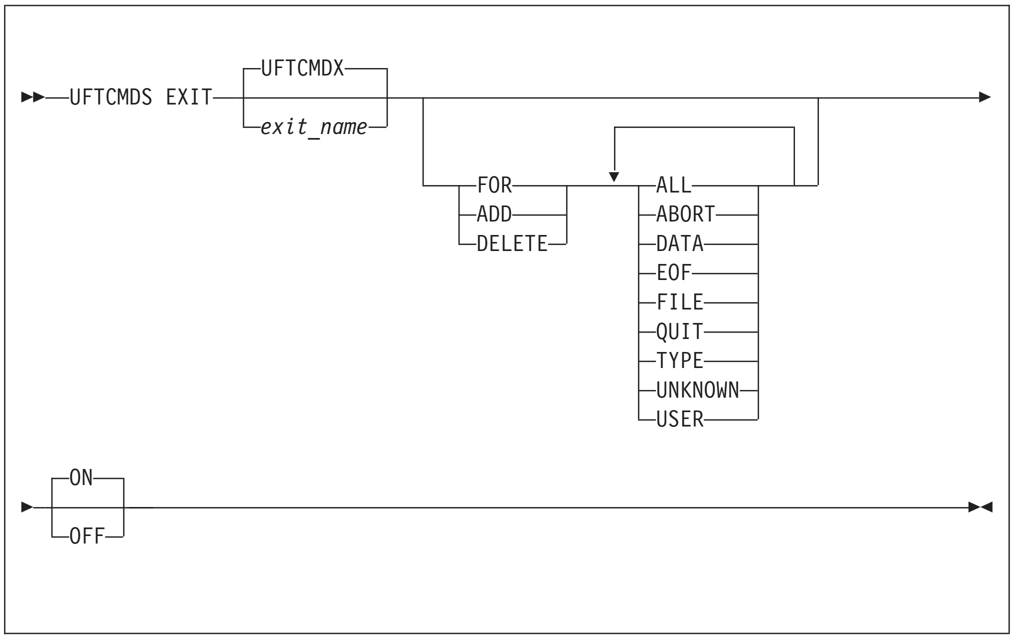
VERBose Initiates tracing of a collection of UFT data, equivalent to specifying CODEflow, CONNectivity and LINE.

NOVERBose Terminates tracing of a collection of UFT data, equivalent to specifying NOCODEflow, NOCONNectivity and NOLINE.

Multiple TRACE subcommands or trace operands may be used to start more than one trace activity.

UFTCMDS EXIT Subcommand

The UFTCMDS subcommand can be used to add or remove protocol command exit points on the running system, or it can be used to enable or disable command exit processing completely. For example, if the UFTCMDX exit exec is active for the DATA and FILE commands, the UFTCMDS EXIT UFTCMDX FOR TYPE ON subcommand will add TYPE to the already active command exit points.



Operands

exit_name Name of the exec to be called to handle any initiated protocol command exits. This exec is called on receipt of any commands for which the exit is enabled. See "Protocol Commands Exit" on page 571 for details on the exit interface. The default exec name is **UFTCMDX**.

FOR Defines which UFT protocol commands are enabled for the exit. If the UFTCMDS exit was previously defined, any commands specified with FOR replace all the previously enabled commands.

ADD Defines which UFT protocol commands are to be added to any existing commands. The enabled commands will be the sum of any previously enabled commands and the commands specified with ADD.

DELETE	Defines which UFT protocol commands are to be deleted from exit processing. The exit remains active for any remaining enabled commands, if any.
command name	The *command name*, which consists of ALL, ABORT, DATA, EOF, FILE, QUIT, TYPE, UNKNOWN, and USER, identifies which protocol commands cause the exit to be called. For any individual command, any valid representation of the command or its synonyms, causes the exit to be called. Two keywords are not commands, but have special meaning. ALL indicates the exec is to be called for all UFT commands eligible for exit processing, and UNKNOWN indicates the exit is to be called for any unknown commands received.
ON	Indicates the exit exec should be invoked for any enabled UFT commands. This is the default.
OFF	Indicates the exit function should be initialized, but not yet invoked, for any UFT protocol commands. A UFTCMDS EXIT ON subcommand is then required to initiate exit processing of UFT protocol commands.

UFT Clients and Servers for Other Platforms

To learn more about UFT clients and servers available for other systems visit the VM TCP/IP homepage at:

`http://www.ibm.com/s390/vm/related/tcpip`

Chapter 26. Configuring the RSCS UFT Client

The RSCS Unsolicited File Transfer (UFT) Client provides support for the UFTASYNC option of the CMS SENDFILE command. This enables the file to be transmitted to the remote system when possible, without tieing up the user's CMS session. The UFT-type link handles *one* file at a time, although successive files may be delivered to different hosts and user IDs. The sample RSCS configuration file has defined eight UFT-type links for handling multiple outgoing UFT data streams.

These steps describe how to configure an RSCS UFT-type link:

RSCS UFT Client Configuration Steps

1. Update the RSCSTCP CONFIG configuration file.
2. Update the RSCSUFT CONFIG configuration file.
3. Update the TCPIP DATA file.

Step 1: Update the RSCSTCP CONFIG Configuration File

The RSCSTCP CONFIG configuration file contains statements you can use to define your RSCS network. This file is read during initialization of the RSCS virtual machine. If this file is not found, RSCS initialization will fail. This file is the main configuration file for the RSCS server and will be stored on the TCP/IP Customization minidisk, TCPMAINT 198. For more information, see "Chapter 15. Configuring the RSCS Print Server" on page 387.

If eight UFT-type links are not enough to handle outgoing client requests, more can be defined by:

- Duplicating LINKDEFINE and PARM statement pairs
- Changing the link name to a unique one
- Adding the link name to the end of the GROUP statement

UFT Client LINKDEFINE and PARM Statements

This section describes the LINKDEFINE and PARM statements necessary in the RSCSTCP CONFIG file for defining an UFT link.

The LINKDEFINE statement defines the default attributes of a single RSCS link. These link attributes apply to the link when it is started.

```
►►──LINKDEFine──UFT──AST──FOrm *──TYPE UFT──────────────────────────────►◄
```

Include as many statements as needed; they are optional. LINKDEFINE statements can be placed anywhere in the configuration file after the LOCAL statement (if this statement exists).

RSCS UFT Client

The UFT keyword is in the *linkid* position, defining the name of the UFT link. This name must be unique on each LINKDEFINE statement. The *linkid* is a 1- to 8-character name of an UFT link that connects your local RSCS server to a remote UFT daemon in a TCP/IP network. The LINK DEFINE statement must come before the PARM statement.

```
                                                    ┌─PORT=608────┐
►►──PARM──UFT──EPARM='C=UFT'──EXIT=UFTXOUT──ITO=0───┤             ├──────────────────►
                                                    └─PORT=portid─┘

   ┌─TCPID=TCPIP─┐
►──┤             ├──────────────────────────────────────────────────────────────────►◄
   └─TCPID=tcpip─┘
```

Operands

UFT
  Specifies the name provided on a LINKDEFINE statement.

PORT=*portid*
  Specifies the port number on the remote host to connect to; defaults to 608 if not specified.

TCPID=*tcpip*
  Specifies the name of the TCPIP virtual machine; defaults to TCPIP if not specified.

Step 2: Update the RSCSUFT CONFIG Configuration File

The RSCS UFT configuration file contains statements you can use to further define how UFT client links handle file processing transmission to a daemon. This file will be stored on the TCP/IP customization minidisk, TCPMAINT 198. This allows you to:

- Change the default table used for EBCDIC to ASCII translation of the data
- Change the default table used for EBCDIC to ASCII translation of UFT commands
- Supply a name used as the owning user ID of the file instead of the user ID of the file originator

An asterisk (*) in column one denotes a comment line. Any line that does not have an asterisk in column one will be interpreted as a configuration entry. All entries must be capitalized.

OWNERNAME=*string*
  Specifies an owning user ID name, up to 32 characters, for the OWNER UFT command. This statement can be used to provide a name other than the name of the file originator.

TOASCII=*string*
  Provides a table for EBCDIC to ASCII translation, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **TOASCII=** records to replace the 256-byte translation table.

TOASCIIC=*string*
Provides a table for EBCDIC to ASCII translation of UFT commands, overriding the default used by the exit. Up to 512 hexadecimal (0-9, A-F) characters may be specified on multiple **TOASCIIC=** records to replace the 256-byte translation table.

Step 3: Update the TCPIP DATA File

The TCPIP data file **must** contain these **uncommented** UFTserverID statements. These two entries enable you to send the files asynchronously using the RSCS server and receive them using the standard UFTD server.

Statement	Description
UFTserverID RSCS	Specifies the user ID the SENDFILE command will spool files to when using the UFTASYNC option. The default of * can be used as long as RSCS is specified in the SYSTEM NETID file.
UFTserverID UFTD	This is used by the PEEK and RECEIVE commands when receiving UFT files to indicate the local UFT server user ID.

For more information on the PEEK, RECEIVE and SENDFILE commands, see *z/VM: CMS Command Reference*.

Chapter 27. Configuring the X.25 Interface

The X25IPI virtual machine runs a Group Control System (GCS) application called X25IPI, which interfaces between the TCPIP virtual machine's IUCV driver and the customer's X.25 network. X25IPI communicates through an IBM 37xx Communication Controller.

Installing the X.25 Interface

To install the X.25 interface, define the X25IPI virtual machine and follow these steps:

X.25 Interface Installation Steps

1. Give X25IPI access to GCS.
2. Define the X25IPI application to the VM/VTAM definition files.
3. Create the X25IPI CONFIG file.
4. Customize the X25IPI GCS file.

Step 1: Give X25IPI Access to GCS

In the GCS configuration file, you must define AUTHUSER NAME=X25IPI. If the GCS Saved Segment is restricted, the X25IPI directory entry must contain a NAMESAVE GCS record. For more information, see the installation guide material supplied with the z/VM product.

Note: X25IPI should not be included in the list of authorized user IDs in the OBEY statement of the PROFILE TCPIP configuration file.

Step 2: Define X25IPI to VM/VTAM

A standard X.25 NPSI GEN configuration operates X25IPI. For more information, see the *X.25 NPSI Planning and Installation* book. The following is a sample X.25 NPSI configuration:

```
        OPTIONS NEWDEFN=YES,USERGEN=X25NPSI
**********************************************************************
APCCU01  PCCU DUMPDS=VTAMDUMP,      DDNAME FOR NCPDUMP DATA SET       *
              CDUMPDS=CSPDUMP,MDUMPDS=MOSSDUMP,                       *
              CONFGDS=CRNCKPT, DDNAME FOR CONFIG RESTART DATA SET     *
              MAXDATA=4096,                                           *
              AUTOSYN=YES,                                            *
              SUBAREA=99,BACKUP=YES,OWNER=A99M,                       *
              NETID=NETA
APCCU02  PCCU DUMPDS=VTAMDUMP,                                        *
              CDUMPDS=CSPDUMP,MDUMPDS=MOSSDUMP,                       *
              CONFGDS=CRNCKPT,                                        *
              MAXDATA=4096,                                           *
              AUTOSYN=YES,                                            *
              SUBAREA=02,BACKUP=YES,OWNER=A02M,                       *
              NETID=NETA
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=VM,          NCP TO BE GENERATED ON  VM       *
                 TWXID=(E8D6E4C3C1D3D311,C2C9C7D5C3D7C3C1D3D325),     *
                 VRPOOL=30,                                           *
                 TRANSFR=32,                                          *
                 NETID=NETA,                                          *
                 X25.USGTIER=5,                                       X
                 X25.IDNUMH=01,                                       X
                 X25.MCHCNT=2,                                        X
                 X25.MAXPIU=64K
         SYSCNTRL OPTIONS=(SESSION,DLRID,DVSINIT,RDEVQ,ENDCALL,MODE,RCN*
                 TRL,RCOND,RECMD,RIMM,XMTLMT,SSPAUSE,NAKLIM,BHSASSC,STORD*
                 SP,BACKUP,LNSTAT,SESINIT)
HOSTA01  HOST MAXBFRU=16,        MINIMUM HOST BUFFER ALLOCATION       *
                 UNITSZ=256,        SIZE OF DATA PORTION OF HOST BUFFER  *
                 SUBAREA=01,                                          *
                 BFRPAD=0            VTAM BUFFER PAD REQUIREMENT (TSC)
*                STATMOD=YES         STATUS MODIFIER
HOSTA02  HOST MAXBFRU=16,                                             *
                 UNITSZ=256,                                          *
                 SUBAREA=02,                                          *
                 BFRPAD=0
*                STATMOD=YES
HOSTA99  HOST MAXBFRU=16,                                             *
                 UNITSZ=256,                                          *
                 SUBAREA=99,                                          *
                 BFRPAD=0
*                STATMOD=YES
*****************************************************************
*     PATH   DECKS                                              *
*****************************************************************
         PATH DESTSA=1,                                               *
                 ER0=(1,1),ER1=(1,1),ER2=(1,1),ER3=(1,1),             *
                 ER4=(1,1),ER5=(1,1),ER6=(1,1),ER7=(1,1),             *
                 VR0=0,VR1=1,VR2=2,VR3=3,VR4=4,VR5=5,                 *
                 VR6=6,VR7=7
         PATH DESTSA=2,                                               *
                 ER0=(2,1),ER1=(2,1),ER2=(2,1),ER3=(2,1),             *
                 ER4=(2,1),ER5=(2,1),ER6=(2,1),ER7=(2,1),             *
                 VR0=0,VR1=1,VR2=2,VR3=3,VR4=4,VR5=5,                 *
                 VR6=6,VR7=7
         PATH DESTSA=99,                                              *
                 ER0=(99,1),ER1=(99,1),ER2=(99,1),ER3=(99,1),         *
                 ER4=(99,1),ER5=(99,1),ER6=(99,1),ER7=(99,1),         *
```

```
              VR0=0,VR1=1,VR2=2,VR3=3,VR4=4,VR5=5,                   *
              VR6=6,VR7=7
        PATH DESTSA=5,                                               *
              ER0=(5,1),ER1=(5,1),ER2=(5,1),ER3=(5,1),               *
              ER4=(5,1),ER5=(5,1),ER6=(5,1),ER7=(5,1),               *
              VR0=0,VR1=1,VR2=2,VR3=3,VR4=4,VR5=5,                   *
              VR6=6,VR7=7
        PATH DESTSA=6,                                               *
              ER0=(5,1),ER1=(5,1),ER2=(5,1),ER3=(5,1),               *
              ER4=(5,1),ER5=(5,1),ER6=(5,1),ER7=(5,1),               *
              VR0=0,VR1=1,VR2=2,VR3=3,VR4=4,VR5=5,                   *
              VR6=6,VR7=7
        PATH DESTSA=7,                                               *
              ER0=(5,1),ER1=(5,1),ER2=(5,1),ER3=(5,1),               *
              ER4=(5,1),ER5=(5,1),ER6=(5,1),ER7=(5,1),               *
              VR0=0,VR1=1,VR2=2,VR3=3,VR4=4,VR5=5,                   *
              VR6=6,VR7=7
        PUDRPOOL NUMBER=10
        LUDRPOOL NUMTYP1=02,NUMTYP2=200,NUMILU=200
**********************************************************************
*                                                                    *
*       NPA GROUP MACRO                                              *
*                            NPACOLL=YES  DEFAULT FOR SDLC           *
**********************************************************************
*A04XNPA  GROUP LNCTL=SDLC,VIRTUAL=YES,NPARSC=YES
*A04NPAL  LINE
*A04NPAP  PU
*A04NPAL1 LU
*A04NPAL2 LU
**********************************************************************
*
*                NPSI DEFINITIONS
*
**********************************************************************
X25XXX   X25.NET CPHINDX=1,                                           X
               NETTYPE=1,                                             X
               DM=YES,                                                X
               OUHINDX=1
         X25.VCCPT INDEX=1,                                           X
               MAXPKTL=128,                                           X
               VWINDOW=2
         X25.OUFT INDEX=1
*********************************************************************
*           DCE STATION
*
*********************************************************************
DCE01    X25.MCH ADDRESS=34,                                          X
               FRMLGTH=131,                                           X
               PKTMODL=8,                                             X
               ANS=CONT,                                              X
               LCGDEF=(0,15),                                         X
               MWINDOW=2,                                             X
               STATION=DCE,                                           X
               SPEED=9600,                                            X
               LCN0=NOTUSED,                                          X
               LLCLIST=(LLC0),                                        X
               GATE=DEDICAT,                                          X
               DBIT=NO,                                               X
               DIRECT=NO,                                             X
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,15),                                           X
               TYPE=S,                                                X
               CALL=INOUT,                                            X
               OUFINDX=1,                                             X
               VCCINDX=1
*********************************************************************
```

```
*             DTE STATION
*
**********************************************************************
DTE01    X25.MCH ADDRESS=36,                                           X
               FRMLGTH=131,                                            X
               PKTMODL=8,                                              X
               ANS=CONT,                                               X
               LCGDEF=(0,15),                                          X
               MWINDOW=2,                                              X
               STATION=DTE,                                            X
               SPEED=9600,                                             X
               LCN0=NOTUSED,                                           X
               LLCLIST=(LLC0),                                         X
               GATE=DEDICAT,                                           X
               DBIT=NO,                                                X
               DIRECT=NO,                                              X
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,15),                                            X
               TYPE=S,                                                 X
               CALL=INOUT,                                             X
               OUFINDX=1,                                              X
               VCCINDX=1
         X25.END
***********************************************************************
***********************************************************************
***********************************************************************
*                                                                     *
*              CHANNEL ADAPTERS                                       *
*                                                                     *
***********************************************************************
***********************************************************************
***** CHANNEL ADAPTER DEFINITIONS                                 *****
***********************************************************************
*
A04XCA1  GROUP LNCTL=CA,CA=TYPE6,NCPCA=ACTIVE,                         *
               TIMEOUT=480.0
***********************************************************************
*            CA ADDR - 00   PHYSICAL PORT 5                           *
***********************************************************************
A04C00   LINE ADDRESS=00,CASDL=00.0,                                   *
               INBFRS=3,DELAY=0.2
A04CP001 PU PUTYPE=5,TGN=1
***********************************************************************
*            CA ADDR - 01   PHYSICAL PORT 6                           *
***********************************************************************
A04C01   LINE ADDRESS=01,CASDL=00.0,                                   *
               INBFRS=3,DELAY=0.2
A04CP011 PU PUTYPE=5,TGN=1
***********************************************************************
*            CA ADDR - 02   PHYSICAL PORT 7                           *
***********************************************************************
A04C02   LINE ADDRESS=02,CASDL=00.0,                                   *
               INBFRS=3,DELAY=0.2
A04CP021 PU PUTYPE=5,TGN=1
***********************************************************************
*            CA ADDR - 03   PHYSICAL PORT 8                           *
***********************************************************************
A04C03   LINE ADDRESS=03,CASDL=00.0,                                   *
               INBFRS=3,DELAY=0.2
A04CP031 PU PUTYPE=5,TGN=1
***********************************************************************
*            CA ADDR - 08    PHYSICAL PORT 1                          *
***********************************************************************
A04C08   LINE ADDRESS=08,CASDL=00.0,                                   *
               INBFRS=3,DELAY=0.2
A04CP081 PU PUTYPE=5,TGN=1
```

```
************************************************************************
*            CA ADDR - 09   PHYSICAL PORT 2                            *
************************************************************************
A04C09   LINE ADDRESS=09,CASDL=00.0,                                    *
               INBFRS=3,DELAY=0.2
A04CP091 PU PUTYPE=5,TGN=1
************************************************************************
*            CA ADDR - 10   PHYSICAL PORT 3                            *
************************************************************************
A04C10   LINE ADDRESS=10,CASDL=00.0,                                    *
               INBFRS=3,DELAY=0.2
A04CP101 PU PUTYPE=5,TGN=1
************************************************************************
*            CA ADDR - 11   PHYSICAL PORT 4                            *
************************************************************************
A04C11   LINE ADDRESS=11,CASDL=00.0,                                    *
               INBFRS=3,DELAY=0.2
A04CP111 PU PUTYPE=5,TGN=1
GENEND   GENEND
```

VTAM Switched Circuit Definition

X.25 NPSI switched virtual circuits (SVCs) appear to VTAM as switched links, requiring a switched circuit definition of a physical unit (PU) and logical unit (LU) for each SVC. 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 VC definitions in the X.25 NPSI configuration. Figure 26 shows the definitions corresponding to the sample X.25 NPSI configuration.

```
         VBUILD TYPE=SWNET,MAXGRP=1,MAXNO=1
VP038001 PU    ADDR=38,IDBLK=003,IDNUM=01002,                         +
               DISCNT=(YES,F),MAXDATA=4096,MAXPATH=1,PUTYPE=1,        +
               SSCPFM=USSNTO
VL038001 LU    LOCADDR=0
```

Figure 26. Sample VTAM Switched Circuit Definition

Note: 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 X25IPI program does not operate correctly.

X25IPI can be attached to the SVC by one of the following methods:

- By hardcoding through the use of the VTAM LOGAPPL statement. For information about the LOGAPPL statement, see *VTAM Resource Definition Reference*.
- By using X.25 NPSI keywords LOGAPPL, CUD0, SUBD, and CTCP. For information about these keywords, see *X.25 NPSI Planning and Installation*.
- By hardcoding the LU names in the VTAM application in the X25IPI DATE application running in the virtual machine.

X25 NPSI Cross-Domain Resources

If you run X25IPI in a different domain from the communication controller running X.25 NPSI, the following describes how the X.25 definitions should be placed:

- The X25IPI application is defined in the domain under which it runs, and is defined as a cross-domain resource in the X.25 NPSI domain.
- Cross-domain resource definitions for the X.25 NPSI MCH PUs and LUs are needed in the X25IPI domain.

- The switched circuits are defined in the X.25 NPSI domain. Cross-domain definitions in the X25IPI domain are not needed.

Step 3: Customize the S25IPI CONFIG File

The X25IPI CONFIG file defines the VTAM and X.25 physical circuits that will be used to allow TCP/IP to communicate between hosts using the X.25 protocol through IBM 37xx communications controllers. A sample configuration file, X25IPI SCONFIG, is shipped on the TCPMAINT 591 disk. It should be copied to the TCPMAINT 198 disk, customized, and renamed to X25IPI CONFIG. This file is used to configure:
- Trace level and debug flags
- VTAM application definition
- X.25 physical circuit definitions
- Buffer specifications
- Timers.

Figure 27 on page 589 is a sample X25IPI configuration file. There is one entry for each line in the file, with a keyword followed by parameter fields separated by one or more blanks. The case is ignored. Comment lines are marked with an asterisk in column 1. The configuration file name should be X25IPI CONFIG.

```
*-- X25IPI Configuration File
*
*-- Trace level (data,off), and debug flags
*                 +         0 display configuration records
*                  +        1 display commands
*                   +       2 trace IUCV events
*                    +      3 trace VTAM events
*                     +     4 display control block addresses
*                      +    5 main loop dispatching
*                       +   6 enable "FOLLOW" trace
*                 0123456789
TRACE    OFF      0000000
*
*-- VTAM Application definition:
*        ApplId   Password
*        -------- --------
VTAM     AX25IPI  X25IPI
*
*-- X.25 Physical circuit definitions:
*        NPSI MCH      DTE address     Window Packet Logical   Incoming
*        LU Name  DNIC (for non-DDN)   Size   Size   Channels  Reserve
*        -------- ---- --------------- -      ----   ---       ---
LINK     XU004    DDN                  2       128   32        0

*
*   Destination address list for this link
*        IP address      X.25 DTE addr      C.U.D.
*        --------------- ---------------    --------
DEST     008
*
*-- Buffer specifications:
*        Datagram Addn'l   VC Queue
*        Max Size Buffers  Limit
*        -------- --       --
Buffers  576      10       5
*
*-- Timers:
*        Idle     Minimum
*        Disconn. Open
*        -------- -------
Timers   180      60
```

Figure 27. Sample X25IPI Configuration File

Keyword	**Description**
Trace	Sets the initial trace and debugging levels.
VTAM	Specifies the application identifier and password to access the VTAM application definition.
Link	Defines each X.25 NPSI MCH to be used. • X.25 NPSI MCH LU name • X.25 data network identifier code for the public network, or DDN for the Defense Data Network • X.25 DTE address for the link, 1 through 15 decimal digits (not needed on DDN Links) • Default window size, in the range 1 through 7 • Default packet size, one of 128, 256, 512, or 1024 • Number of logical channels subscribed • Number of logical channels reserved for incoming calls.
Dest	Specifies the destination address.

- The IP address, in dotted-decimal form. One byte must be supplied; omitted trailing bytes are not checked when determining a match.
- The corresponding X.25 DTE address.
- The call user data (protocol identifier) to be used, in hexadecimal notation. The standard value is X'CC'.
- The DTE address and call user data fields should be omitted on DDN links to obtain standard address calculation.

Buffers Defines the buffers.

- Size of buffers to use for datagrams; the minimum is 576. This value must match the TCPIP GATEWAY *max_packet_size.*
- Number of buffers to allocate, in addition to the minimum of two for each logical channel.
- Limit on number of buffers queued outbound on any single virtual circuit.

Timers Specifies the amounts of time.

- Time, in seconds, after which an inactive connection is cleared.
- Minimum time, in seconds, a connection is held, before it can be preempted for a new destination.

Step 4: Customize X25IPI GCS

Figure 28 shows the content of a sample X25IPI GCS file (supplied on the TCPMAINT 591 disk as file X25IPI SAMPGCS). Your customized file should be stored on the TCPMAINT 198 disk as file X25IPI GCS. Within your customized file, update parameters and variable assignments to reflect appropriate values for your environment. In addition, add a `CP SPOOL CONSOLE` statement to your X25IPI GCS file if a user ID other than TCPMAINT is to receive the console log. If additional X.25 servers are required, copy the X25IPI GCS file to TCPMAINT 198, appropriately naming each copied file as *userid* GCS.

```
/**/
x25cmd = "X25"                                /* Set command name      */
Parse arg parm                                /* If there are any      */
If parm <> "" Then                            /* parms, then this      */
  Do                                          /* is a command, not     */
    x25cmd parm                               /* a profile call.       */
    Exit rc                                   /* Invoke command and    */
  End                                         /* exit.                 */

Config  = "X25IPI CONFIG *"                   /* Configuration file    */
TcpData = "TCPIP DATA *"                      /* Standard client data  */

"FILEDEF X25IPI   DISK" Config
"FILEDEF X25IPITD DISK" TcpData

"GLOBAL LOADLIB XNX25"
"LOADCMD x25cmd "XNX25CMD"
x25cmd "START XNX25IPI"
Exit rc
```

Figure 28. Sample X25IPI GCS File

Note: XNX25 is the name of the LOADLIB, which contains the link-edited XNX25IPI and XNX25CMD modules. X25IPI is automatically started by X25IPI GCS when GCS is initialized.

X25IPI Commands

You can enter commands on the GCS service machine console; however, you must prefix the commands with the program name specified to LOADCMD in the X25IPI GCS file (shown as X25IPI). The following is the syntax for the X25IPI commands.

```
►►──LIST───────────────────────────────────────────────────────────►◄
```

```
►►──TRAFFIC────────────────────────────────────────────────────────►◄
```

```
►►──RESTART──┬───────┬─────────────────────────────────────────────►◄
             └─mchlu─┘
```

```
►►──TRACE──┬─id─┬──┬─DATA─┬────────────────────────────────────────►◄
           └─*──┘  └─OFF──┘
```

```
►►──DEBUG──digits──────────────────────────────────────────────────►◄
```

```
►►──SNAP──┬────┬───────────────────────────────────────────────────►◄
          └─id─┘
```

```
►►──EVENTS──┬────┬─────────────────────────────────────────────────►◄
            └─id─┘
```

```
►►──HALT───────────────────────────────────────────────────────────►◄
```

```
►►──CANCEL─────────────────────────────────────────────────────────►◄
```

X.25 Interface

The following are the commands that you can enter on the GCS service machine console:

Command	Description
LIST	Displays a list of virtual circuit status.
TRAFFIC	Displays traffic counts.
RESTART *mchlu*	Attempts to reacquire failed links (MCHs), after reactivating them with VTAM. Where *mchlu* is an optional LU name from a link definition.
TRACE *id* **DATA** | **OFF**	Alters data trace level. Where *id* is an LU name, logon ID, or asterisk. DATA indicates to scan trace and OFF indicates not to trace.
DEBUG *digits*	Alters debug settings. Where *digits* is a string of debug levels, as in the CONFIG file.
SNAP *id*	Displays program data areas, for debugging. Where *id* is an optional LU name or logon ID.
EVENTS *id*	Displays event handler names, for debugging. Where *id* is an optional LU name or logon ID.
HALT	Terminates the program, closing all connections.
CANCEL	Cancels the X25IPI program task, producing a dump.

Chapter 28. X Window System Graphical Data Display Manager Support

This chapter describes how to install the X Window System GDDM interface support provided with TCP/IP.

X Window System GDDM Support

The X Window System GDDM interface permits graphics display output from the IBM Graphical Data Display Manager (GDDM) to be displayed on workstations that support the X Window System.

The X Window System GDDM interface translates the data stream, created by GDDM, to the X Window System protocol, and transmits it by TCP/IP to the X Window System server for the workstation display.

Note: If the X Window System GDDM interface is installed on your system, but is not active, GDDM transmits data as if the interface is not present.

X Window System GDDM Interface Files

The following is a list of the X Window System GDDM interface files:

File Name	Description
GDDMXD TXTLIB	Executable code
INSTGDXD EXEC	Installation EXEC
GDDMXD EXEC	Activates and deactivates GDDMXD virtual machine
X DEFAULTS	Optional X Window sample defaults file
GDXALTCS PSS	Alternative character set
GDXAPLCS SAMPMAP	Sample keyboard mapping for the APL2 character set
KEYCODE TEXT	Sample text file for displaying keycodes

Installing the X Window System GDDM Interface

The X Window System GDDM Interface support should be installed by the person responsible for installation and maintenance of GDDM/VM on your system, so that the modules built using this procedure can be placed into production using the appropriate measures and minidisks.

Required Files and TXTLIBs

The following files and TXTLIBs are required during the installation of the X Window System GDDM interface. Verify that these files are on accessible minidisks before you invoke INSTGDXD EXEC.

File Name	Contents
ADMGLIB TXTLIB	GDDM
ADMNLIB TXTLIB	GDDM
ADMPLIB TXTLIB	GDDM/PGF (if GDDM/PGF is installed)

X Window System Graphical Data Display Manager Support

COMMTXT TXTLIB	Common TCP/IP Text Library
GDDMXD TXTLIB	GDDMXD/VM
SCEELKED TXTLIB	IBM Language Environment (Runtime Library)
INSTGDXD EXEC	GDDMXD/VM
X11LIB TXTLIB	X Window Interface

X Window System GDDM Interface Installation Steps

Use the INSTGDXD EXEC and follow the steps below to install the X Window System GDDM interface:

1. Invoke the INSTGDXD EXEC.

   The INSTGDXD EXEC performs the following:

   a. Verifies that all required files are available.

   b. Builds the GDXLIOX0 MODULE.

      The GDXLIOX0 module is loaded during GDDMXD execution.

   c. Builds the demonstration modules, GXDEMO1 through GXDEMO6.

   d. Builds the KEYCODE MODULE.

   If errors were reported by the INSTGDXD EXEC, resolve any problem before continuing with the installation.

2. Copy the following files to the GDDM library minidisks for general use:
   - GDXLIOX0 MODULE
   - GDXCLOSE MODULE
   - KEYCODE MODULE
   - GXDEMO1 MODULE
   - GXDEMO2 MODULE
   - GXDEMO3 MODULE
   - GXDEMO4 MODULE
   - GXDEMO5 MODULE
   - GXDEMO6 MODULE

Chapter 29. Using Translation Tables

TCP/IP uses translation tables to convert transmitted data between EBCDIC and ASCII. Because the meanings of the terms “EBCDIC” and “ASCII” depend on the particular operating system and the national language (English, French, etc.) being used on a particular system, TCP/IP provides many different translation tables to meet the diverse needs of VM/ESA users.

In addition to the more than 200 translations provided by IBM, you can create custom tables to meet your specific requirements.

The following sections provide the information you need to understand what translation tables are, how they are used by TCP/IP applications, and how you can create your own custom translations.

Character Sets and Code Pages

When you display or print a document, you see a collection of characters or symbols. A group of characters or symbols taken together and treated as a single entity is called a **character set**. A character set may contain hundreds or even thousands of characters.

In a Single-Byte Character Set (SBCS), one 8-bit byte is used to represent a single character. This means there are only 256 possible bit patterns or **code points** available to represent a character. All Western languages can be represented by an SBCS character set.

A Double-Byte Character Set (DBCS) uses *two* bytes to represent a single character, providing a theoretical maximum of 65536 characters. In practice, DBCS character sets contain far fewer than 65536 characters. Eastern languages such as Japanese Kanji, Korean Hangeul, and traditional Chinese require a DBCS character set.

A collection of all of 256 (for SBCS) or 65536 (for DBCS) code points and their corresponding individual character assignments are called a **code page**.

While it is true that always using a universal DBCS character set such as Unicode would eliminate the need to perform EBCDIC-ASCII translation, most of the operating systems and standard TCP/IP application protocols in use today were developed before the advent of DBCS. As a consequence, every country or common geographic region developed its own country-specific SBCS code page, particularly in the EBCDIC environment. Characters were deleted, added, and their order changed.

Consequently, it is necessary to understand and manage the use of code pages. To assist in that effort, IBM has assigned a unique number to many of the EBCDIC and ASCII code pages you will use. The specific code page translations provided with TCP/IP are listed in Table 36 on page 598. Facilities are provided so that you may supplement the translations provided by IBM with your own.

The TCP/IP translation tables convert data from one code page to another, so the table you choose depends on the code pages being used by the systems involved and your knowledge of how a file was created.

Using Translation Tables

It is important to recognize that changing the default translation table for servers such as FTP and NFS can corrupt data in a file if that file is uploaded and downloaded using different translation tables. (This does not apply to binary transfers, of course.)

TCP/IP Translation Table Files

TCP/IP translation tables are machine-readable binary files that are usually kept on the TCP/IP user disk, TCPMAINT 592.

Most of these files are provided by IBM, and others may be created by compiling SBCS or DBCS translation table source files using the CONVXLAT command, described in "Converting Translation Tables to Binary" on page 604.

Table 34. Translation Table Files

Character Set	Language	Source File Type	Table File Type
SBCS	Any	TCPXLATE	TCPXLBIN
DBCS	Japanese Kanji	TCPKJLAT	TCPKJBIN
DBCS	Korean Hangeul	TCPHGLAT	TCPHGBIN
DBCS	Traditional Chinese	TCPCHLAT	TCPCHBIN

Note that the file types for the different languages are different. There can be up to four translation table that have the same file name – one for all SBCS languages, and one for each of the three DBCS languages.

SBCS tables contain translations for one pair of code pages. DBCS tables may contain multiple translations.

To modify an IBM-provided translation table:
1. Modify the source file as required
2. Run the CONVXLAT program, specifying the modified source as input
3. Copy the resulting TCP*xx*BIN file to the TCP/IP user disk, TCPMAINT 592. Translation tables made available to servers should also be made available to clients.

To create a new translation table, first copy an existing source file and then follow the procedure outlined above.

SBCS translation tables may be read by CMS applications using the DTCXLATE CSL routine. For more information on DTCXLATE, see the *z/VM: CMS Callable Services Reference*.

Translation Table Search Order

Most TCP/IP client and server programs provide a way for the you to change the translation table that is used. This is done using either client command line options, server initialization parameters, or configuration file statements. For example, the CMS FTP client provides a **TRANSLATE** option that you can use to provide the name of a translation table.

If no such option or configuration statement is provided, the clients and servers will use a *preferred* translation table that is specific to a particular client or server. If the preferred table cannot be found, the common *standard* translation will be used. The standard translation is loaded from STANDARD TCP*xx*BIN, if it is available, or from an equivalent that is compiled into the program.

Table 35 shows the option or configuration statement that may be used to provide the name of a translation table to be used by each client or server. Any program not listed can be assumed to use STANDARD.

Table 35. Preferred Translation Tables

Program	Option	Preferred Translation Table
SMTP Server	None[1]	SMTP
FTP Server	SITE TRANSLATE *table_name*[2]	SRVRFTP
FTP Client	TRANSLATE *table_name*[2]	FTP
UFT Client (SENDFILE)	TRANSLATE *table_name*[2]	STANDARD
UFT Server	TRANSLATE *table_name*[2]	STANDARD
LPR Client	TRANSLATE *table_name*[2]	LPR
LPD Server	TRANSLATETABLE *table_name*[2]	LPD
NFS Server	XLATE=*table_name*[2]	VMNFS
TELNET Client	TRANSLATE *table_name*[2]	TELNET
TELNET Server (line mode)	None	STLINMOD
TFTP Client	TRANSLATE *table_name*[2]	TFTP

Note[1]: For SMTP, an additional translation table may be specified for 8-bit MIME support. The *TCP/IP Planning and Customization* contains more information in the “8BITMIME Statement” section.

Note[2]: *table_name* is the file name of a TCP/IP translation table.

If a table is explicitly referenced by a program option or configuration statement, the program will display an error message and stop if the translation table file cannot be found or loaded.

The file type of the translation table depends on whether any DBCS features are used. If Kanji translation is requested, the file type is TCPKJBIN. If Korean translation is requested, the file type is TCPHGBIN. For traditional Chinese, the file type is TCPCHBIN.

The *TCP/IP User's Guide* contains information on the TRANSLATE option for the TCP/IP clients, and the *TCP/IP Planning and Customization* contains information for the servers. See the *z/VM: CMS Command Reference* for information about the SENDFILE command.

Special Telnet Requirements

Telnet Client

The Telnet client requires a translation table that is different from the default table, STANDARD. The preferred translation table is provided by IBM as TELNET TCPXLBIN. If this file is not found, however, the default table will be used.

Country-specific translation tables for the Telnet application are provided. These tables have the file type TELXLATE. You must rename the selected file to TELNET TCPXLATE before it is converted to binary using the CONVXLAT command. The resulting TELNET TCPXLBIN should then be copied to TCPMAINT 592, replacing the IBM version.

Using Translation Tables

Note: You cannot use the Telnet translation tables to change the LineFeed (X'0A') character.

Telnet Server

For line mode Telnet sessions, translation is performed by the Telnet server using the STANDARD translation table. If this table does not meet your needs, you can create an STLINMOD TCPXLATE table, convert it to binary using CONVXLAT, and copy the resulting STLINMOD TCPXLBIN file to the TCP/IP customization disk, TCPMAINT 198.

IBM-Supplied Translation Tables

In order to meet the translation needs of users and installations worldwide, more than 200 translation tables are provided with TCP/IP for your use. Some tables already exist in binary form; others require conversion using the CONVXLAT command, as described in "Converting Translation Tables to Binary" on page 604.

Table 36. IBM Translation Tables

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
United States and Canada	0037*rrrr*	TCPXLATE	37	*rrrr*
Austria and Germany	0273*rrrr*	TCPXLATE	273	*rrrr*
Denmark and Norway	0277*rrrr*	TCPXLATE	277	*rrrr*
Finland and Sweden	0278*rrrr*	TCPXLATE	278	*rrrr*
Italy	0280*rrrr*	TCPXLATE	280	*rrrr*
Spain and Spanish-speaking Latin America	0284*rrrr*	TCPXLATE	284	*rrrr*
United Kingdom	0285*rrrr*	TCPXLATE	285	*rrrr*
France	0297*rrrr*	TCPXLATE	297	*rrrr*
International	0500*rrrr*	TCPXLATE	500	*rrrr*
Iceland	0871*rrrr*	TCPXLATE	871	*rrrr*
ISO 8859-15	0924*rrrr*	TCPXLATE	924	*rrrr*
OpenEdition® (POSIX)	1047*rrrr*	TCPXLATE	1047	*rrrr*
United States and Canada (Euro)	1140*rrrr*	TCPXLATE	1140	*rrrr*
Austria and Germany (Euro)	1141*rrrr*	TCPXLATE	1141	*rrrr*
Denmark and Norway (Euro)	1142*rrrr*	TCPXLATE	1142	*rrrr*
Finland and Sweden (Euro)	1143*rrrr*	TCPXLATE	1143	*rrrr*
Italy (Euro)	1144*rrrr*	TCPXLATE	1144	*rrrr*
Spain and Spanish-speaking Latin America (Euro)	1145*rrrr*	TCPXLATE	1145	*rrrr*
United Kingdom (Euro)	1146*rrrr*	TCPXLATE	1146	*rrrr*
France (Euro)	1147*rrrr*	TCPXLATE	1147	*rrrr*
International (Euro)	1148*rrrr*	TCPXLATE	1148	*rrrr*
Iceland (Euro)	1149*rrrr*	TCPXLATE	1149	*rrrr*
OpenEdition	1047*rrrr*	TCPXLATE	1047	*rrrr*
ISO 8859-15 (EBCDIC)	0924*rrrr*	TCPXLATE	924	*rrrr*
ISO 8859-15 (ASCII)	*hhhh*0923	TCPXLATE	*hhhh*	923

Table 36. IBM Translation Tables (continued)

Country	Translation Table File Name	Translation Table File Type	Host Code Page	Remote Code Page
OS/2	*hhhh*0850	TCPXLATE	*hhhh*	850
OS/2 (Euro)	*hhhh*0858	TCPXLATE	*hhhh*	858
ISO 8859-1 (ASCII)	*hhhh*0819	TCPXLATE	*hhhh*	819
Microsoft® Windows®	*hhhh*1252	TCPXLATE	*hhhh*	1252
Austria and Germany	AUSGER	TCPXLATE	273	850
Belgium	BELGIAN	TCPXLATE	500	850
Canada	CANADIAN	TCPXLATE	37	850
Denmark and Norway	DANNOR	TCPXLATE	277	850
Netherlands	DUTCH	TCPXLATE	37	850
Finland and Sweden	FINSWED	TCPXLATE	278	850
France	FRENCH	TCPXLATE	297	850
Italy	ITALIAN	TCPXLATE	280	850
Japan	JAPANESE	TCPXLATE	281	850
OpenEdition	POSIX	TCPXLATE	1047	819
Portugal	PORTUGUE	TCPXLATE	37	850
Spain and Spanish-speaking Latin America	SPANISH	TCPXLATE	284	850
Switzerland (French)	SWISFREN	TCPXLATE	500	850
Switzerland (German)	SWISGERM	TCPXLATE	500	850
United Kingdom	UK	TCPXLATE	285	850
United States	US	TCPXLATE	37	850
Standard (SBCS)	STANDARD	TCPXLATE	EBCDIC	ASCII
Standard Japanese Kanji	STANDARD	TCPKJLAT		
JIS X0208 1978			300	X0208
JIS X0208 1983			300	X0208
Shift JIS X0208			300	X0208
Extended Unix Code			300	EUC
IBM			300	300
Standard Korean Hangeul	STANDARD	TCPHGLAT		
KSC 5601 SBCS			833	1088
KSC 5601 DBCS			834	951
Hangeul SBCS			833	891
Hangeul DBCS			834	926
Standard Traditional Chinese	STANDARD	TCPCHLAT		
Traditional Chinese SBCS			037	904
Traditional Chinese DBCS			835	927

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

Notes:

1. STANDARD TCPXLBIN is a 7-bit non-reversible translation table. For inbound data, all ASCII characters with values in the range X'80'-X'FF' will have the same translation as values X'00'-X'7F'. The high-order bit of each ASCII byte is

ignored and is assumed to be zero. For example, ASCII X'B1' and X'31' will both be converted to EBCDIC X'F1'. For outbound data, EBCDIC control characters that do not have ASCII equivalents are converted to ASCII X'1A'. STANDARD should be used in situations where a client or server is known to treat the high-order bit in each byte as a parity bit.

2. All other SBCS translation tables provide a unique, one-to-one mapping of all 256 code points.
3. All tables translate ASCII LineFeed (LF, X'0A') to and from EBCDIC LF (X'25'), except for POSIX and 1047*rrrr*, which use EBCDIC NewLine (NL, X'15') instead.
4. Host code pages 924 and 1140-1149 include translations for the euro currency symbol.

Customizing SBCS Translation Tables

All SBCS translation table files contain two tables. The first table is used to translate from ASCII to EBCDIC. The second table is used to translate from EBCDIC to ASCII.

To read the translation tables, find the row for the first hex digit (1) and the column for the second hex digit (2). The point where the row and column intersect is the translation value.

For example, to find the EBCDIC translation for the ASCII character X'A7', find row A0 (3) and column 07 (4) in the following example. The point where row A0 and column 07 intersect shows a value of X'7D', so the ASCII character X'A7' will be translated to a X'7D' in EBCDIC. To customize the translation table, alter the translate value where the row and column intersect to the new value.

```
;
; ASCII-to-EBCDIC table
;                       4                       2
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;                                                        1
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 00 ;
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 10 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; 20 ;
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; 30 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; 40 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; 50 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; 60 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; 70 ;
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 80 ;
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 90 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; A0 ;  3
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; B0 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; C0 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; D0 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; E0 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; F0 ;
;
; EBCDIC-to-ASCII table
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;
  00 01 02 03 1A 09 1A 7F 1A 1A 1A 0B 0C 0D 0E 0F    ; 00 ;
  10 11 12 13 1A 1A 08 1A 18 19 1A 1A 1C 1D 1E 1F    ; 10 ;
  1A 1A 1C 1A 1A 0A 17 1B 1A 1A 1A 1A 1A 05 06 07    ; 20 ;
  1A 1A 16 1A 1A 1E 1A 04 1A 1A 1A 1A 14 15 1A 1A    ; 30 ;
  20 A6 E1 80 EB 90 9F E2 AB 8B 9B 2E 3C 28 2B 7C    ; 40 ;
  26 A9 AA 9C DB A5 99 E3 A8 9E 21 24 2A 29 3B 5E    ; 50 ;
  2D 2F DF DC 9A DD DE 98 9D AC BA 2C 25 5F 3E 3F    ; 60 ;
  D7 88 94 B0 B1 B2 FC D6 FB 60 3A 23 40 27 3D 22    ; 70 ;
```

```
F8 61 62 63 64 65 66 67 68 69 96 A4 F3 AF AE C5    ; 80 ;
8C 6A 6B 6C 6D 6E 6F 70 71 72 97 87 CE 93 F1 FE    ; 90 ;
C8 7E 73 74 75 76 77 78 79 7A EF C0 DA 5B F2 F9    ; A0 ;
B5 B6 FD B7 B8 B9 E6 BB BC BD 8D D9 BF 5D D8 C4    ; B0 ;
7B 41 42 43 44 45 46 47 48 49 CB CA BE E8 EC ED    ; C0 ;
7D 4A 4B 4C 4D 4E 4F 50 51 52 A1 AD F5 F4 A3 8F    ; D0 ;
5C E7 53 54 55 56 57 58 59 5A A0 85 8E E9 E4 D1    ; E0 ;
30 31 32 33 34 35 36 37 38 39 B3 F7 F0 FA A7 FF    ; F0 ;
```

Syntax Rules for SBCS Translation Tables

- Blanks are used only as delimiters for readability purposes.
- Comments may be included, either on a separate line or at the end of a line. Comments must start with a semicolon (;).

Customizing DBCS Translation Tables

Each DBCS translation table file contains more than one translation table. TCPHGLAT and TCPHGBIN, for example, contain EBCDIC to ASCII and ASCII to EBCDIC translation tables for both the KSC 5601 and Hangeul PC code pages.

The standard DBCS binary tables are used by the FTP server, SMTP server, and FTP client programs.

The figures on the following pages show examples of the standard source for the Kanji, Hangeul, and Traditional Chinese DBCS translation tables.

These source files contain two column pairs for each code page. The first column pair specifies double-byte EBCDIC to ASCII code point mappings for the indicated code page. The second column pair specifies double-byte ASCII to EBCDIC code point mappings for the indicated code page.

Existing code point mappings may be changed by simply overwriting the existing hexadecimal code. New code point mappings may be specified by adding a new column pair with two double-byte hexadecimal codes. Code point mappings that are not specified, and are within the valid range for the code page, default to the “undefined” character, X'FFFF'.

The source file format allows EBCDIC to ASCII and ASCII to EBCDIC mappings to be specified separately. When adding or changing a code point mapping, care should be taken to modify both mappings for the code point. If, for example, a new mapping is added for EBCDIC to ASCII only, the ASCII to EBCDIC mapping for that code point will be the “undefined” character.

Any new code point mappings added outside the valid range for the corresponding code page will not be used by the programs that load the binary table.

DBCS Translation Table

The DBCS translation tables also contain SBCS code point mappings. These are used for mixed-mode DBCS strings, containing both SBCS and DBCS characters. Shift-out (X'0E') and shift-in (X'0F') characters are used on the EBCDIC host to denote the beginning and end of DBCS characters within a mixed-mode string.

The DBCS source files must contain exactly 256 SBCS code point mappings, situated at the end of the table. These may be modified to contain the required hexadecimal value.

Syntax Rules for DBCS Translation Tables

- Comments may be included, either on a separate line or at the end of a line. Comments must start with a semicolon (;).
- Code point mappings in the file are position dependent. The first non-comment line for the DBCS and SBCS tables in the file will be used to establish the column position of the code point mappings, and must contain a conversion pair for each code page. Any conversion pairs on following lines must use the same column positions.
- It is permissible to leave blanks for code point mappings after the first line in the DBCS and SBCS areas. For example, if a line contains only one conversion pair, the column position will be used to determine which code page it refers to.
- The first column of each code page column pair, the “code index”, must be in ascending numerical order. Any gaps in the ascending order will be marked as “undefined” in the binary table created by CONVXLAT.

Sample DBCS Translation Tables

The following examples are from the STANDARD DBCS translation table source files. Because these files are very large, only excerpts from the tables are shown. Ellipses (...) are used to indicate that information has been deleted.

Japanese Kanji DBCS Translation Tables

```
;
; STANDARD TCPKJLAT - Japanese translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
; Use CONVXLAT to generate STANDARD TCPKJBIN
; from this source file.
;
;DBCS Area - SJISETA,SJISATE
;          - JDECETA,JDECATE not used for STANDARD TCPKJBIN generation.
;
;SJISETA    SJISATE     JIS78ETA   JIS78ATE    JIS83ETA   JIS83ATE    ...
;
 4040 8140  8140 4040   4040 2121  2121 4040   4040 2121  2121 4040   ...
 4141 83BF  8141 4344   4141 2641  2122 4344   4141 2641  2122 4344   ...
 4142 83C0  8142 4341   4142 2642  2123 4341   4142 2642  2123 4341   ...
 4143 83C1  8143 426B   4143 2643  2124 426B   4143 2643  2124 426B   ...
 4144 83C2  8144 424B   4144 2644  2125 424B   4144 2644  2125 424B   ...
 4145 83C3  8145 4345   4145 2645  2126 4345   4145 2645  2126 4345   ...
 4146 83C4  8146 427A   4146 2646  2127 427A   4146 2646  2127 427A   ...
 4147 83C5  8147 425E   4147 2647  2128 425E   4147 2647  2128 425E   ...
 4148 83C6  8148 426F   4148 2648  2129 426F   4148 2648  2129 426F   ...
 4149 83C7  8149 425A   4149 2649  212A 425A   4149 2649  212A 425A   ...
⋮


;
; SBCS Area
;
;---------TCPKJBIN generation (no codefiles)--------------| |--------------
;SJISETA ATE   JIS78ETA ATE    JIS83ETA ATE   SJEUCETA ATE   J7KETA J7KATE
;
 00 00  00 00   00 00  00 00   00 00  00 00   00 00  00 00   00 00  00 00  ..
 01 01  01 01   01 01  01 01   01 01  01 01   01 01  01 01   01 01  01 01  ..
 02 02  02 02   02 02  02 02   02 02  02 02   02 02  02 02   02 02  02 02  ..
 03 03  03 03   03 03  03 03   03 03  03 03   03 03  03 03   03 03  03 03  ..
 04 1A  04 37   04 1A  04 37   04 1A  04 37   04 1A  04 37   04 1A  04 37  ..
 05 09  05 2D   05 09  05 2D   05 09  05 2D   05 09  05 2D   05 09  05 2D  ..
 06 1A  06 2E   06 1A  06 2E   06 1A  06 2E   06 1A  06 2E   06 1A  06 2E  ..
 07 7F  07 2F   07 7F  07 2F   07 7F  07 2F   07 7F  07 2F   07 7F  07 2F  ..
 08 1A  08 16   08 1A  08 16   08 1A  08 16   08 1A  08 16   08 1A  08 16  ..
 09 1A  09 05   09 1A  09 05   09 1A  09 05   09 1A  09 05   09 1A  09 05  ..
⋮
```

Hangeul DBCS Translation Tables

```
;
; STANDARD TCPHGLAT - Korean translation tables.
;
; ETA = Ebcdic to Ascii Conversion (Host - PC)
; ATE = Ascii to Ebcdic Conversion (PC - Host)
;
;use CONVXLAT to generate STANDARD TCPHGBIN
; from this source file.
;
; DBCS Area
;
; Code Page ID - 951      Code Page ID - 926
; KSCETA     KSCATE        HANETA     HANATE
;
  4040 A1A1  8FA1 D541     4040 8140  8140 4040
  4141 A1A2  8FA2 D542     4141 8141  8141 4141
  4142 A1A3  8FA3 D543     4142 8142  8142 4142
  4143 A1A4  8FA4 D544     4143 8143  8143 4143
  4144 A1A5  8FA5 D545     4144 8144  8144 4144
  4145 A1A6  8FA6 D546     4145 8145  8145 4145
  4146 A1A7  8FA7 D547     4146 8146  8146 4146
  4147 A1A8  8FA8 D548     4147 8147  8147 4147
  4148 A1A9  8FA9 D549     4148 8148  8148 4148
  4149 A1AA  8FAA D54A     4149 8149  8149 4149
⋮

;
; SBCS Area
;
;Code Page ID 1088   Code Page ID 891
;SKSCETA   SKSCATE   SHANETA   SHANATE
;
 00 00     00 00     00 00     00 00
 01 01     01 01     01 01     01 01
 02 02     02 02     02 02     02 02
 03 03     03 03     03 03     03 03
 04 FF     04 37     04 FF     04 37
 05 09     05 2D     05 09     05 2D
 06 FF     06 2E     06 FF     06 2E
 07 1C     07 2F     07 1C     07 2F
 08 FF     08 16     08 FF     08 16
 09 FF     09 05     09 FF     09 05
⋮
```

Traditional Chinese DBCS Translation Tables

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

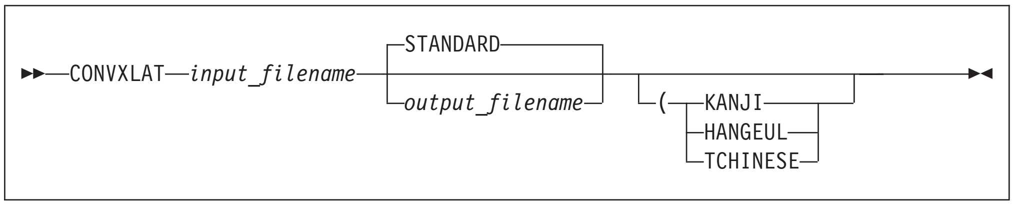
```
 4141 83BF   8141 4344
 4142 83C0   8142 4341
 4143 83C1   8143 426B
 4144 83C2   8144 424B
 4145 83C3   8145 4345
 4146 83C4   8146 427A
 4147 83C5   8147 425E
 4148 83C6   8148 426F
 4149 83C7   8149 425A
⋮

;
; SBCS Area
;
; STCHETA  STCHATE
;
 00  00   00  00
 01  01   01  01
 02  02   02  02
 03  03   03  03
 04  cf   04  37
 05  09   05  2d
 06  d3   06  2e
 07  7f   07  2f
 08  d4   08  16
 09  d5   09  05
⋮
```

Converting Translation Tables to Binary

The CONVXLAT command converts a translation table source file to a binary file that usable by TCP/IP client and server programs. CONVXLAT may be used to convert both SBCS and DBCS source file.

The CONVXLAT Command

The syntax of the CONVXLAT command is:



The parameters of the CONVXLAT command are:

Parameter	**Description**
input_filename	Specifies the file name of the source file to be converted. The source file must have a file type of TCPXLATE for SBCS tables, or a file type of TCPKJLAT, TCPHGLAT, or TCPCHLAT for DBCS tables. The first file with the required file name and file type in the standard minidisk search order is used.
output_filename	Specifies the destination file name created by the conversion. If this parameter is not specified, it defaults to the name STANDARD. The destination file type will be TCPXLBIN for SBCS tables, or TCPKJBIN, TCPHGBIN, or TCPCHBIN for DBCS tables. The destination file mode is A, which must be accessed in write mode.
KANJI	Specifies that the table being converted is the Kanji DBCS/SBCS

translation table. The file type of the source file must be TCPKJLAT. The file type of the destination file will be TCPKJBIN.

HANGEUL	Specifies that the table being converted is the Hangeul DBCS/SBCS translation table. The file type of the source file must be TCPHGLAT. The file type of the destination file will be TCPHGBIN.
TCHINESE	Specifies that the table being converted is the Traditional Chinese DBCS/SBCS translation table. The file type of the source file must be TCPCHLAT. The file type of the destination file will be TCPCHBIN.

If no optional parameters are specified, then *input_filename* is assumed to contain an SBCS translation table.

Chapter 30. Testing

This section describes the testing functions available in TCP/IP Level 3A0.

Loopback Testing

Loopback testing lets you know if a server is online. It is used by FTP, Telnet, SMTP, and Portmap.

TCP/IP Checksum Testing

TCP/IP uses a checksum to detect errors in a data packet. A checksum is an integer added to the packet that is calculated from the data portion of the packet. TCP/IP calculates the checksum and then verifies it against the checksum value in the packet. If the two values do not match, the packet is rejected.

The NOCHECKSUM statement turns off checksum verification. This statement should only be used for debugging purposes. For example, you can use NOCHECKSUM to find problems with a noisy network, or to find a gateway or router that is losing bits in the transmission.

CHECKSUM Statement

The CHECKSUM statement instructs the TCPIP virtual machine to reenable TCP checksum testing on incoming messages, if it has been disabled by the NOCHECKSUM statement.

```
►►──CHECKSUM──────────────────────────────────────────────────────────►◄
```

The TCP/IP module's default configuration is to verify checksums, so the CHECKSUM statement is rarely needed.

The CHECKSUM statement has no operands.

NOCHECKSUM Statement

The NOCHECKSUM statement instructs the TCPIP virtual machine to ignore TCP checksum errors on incoming datagrams. Checksums are generated for outgoing datagrams, regardless of whether the NOCHECKSUM statement is used.

NOCHECKSUM should be used only for debugging purposes. It can affect data integrity, if used for normal operations.

```
►►──NOCHECKSUM────────────────────────────────────────────────────────►◄
```

To restore checksum validation, use the CHECKSUM statement.

The NOCHECKSUM statement has no operands.

TCP/IP Datagram Dropping

TCP/IP can be configured to automatically *lose* packets to force programs to retransmit data. This is especially useful for determining a program's ability to recover from lost or misdirected packets.

The DROP statement allows you to determine the percentage that VM is deliberately loosing or discarding. This command should only be used for testing, as it decreases performance.

DROP Statement

The DROP statement is useful for testing transmission and acknowledgment strategies by simulating network errors. This statement defines the percentage of outgoing datagrams that IP discards.

```
           ┌─0──────────┐
►►──DROP───┴─percentage─┴──────────────────────────────────────►◄
```

Operands

percentage	Specifies an integer from 0 to 100. DROP 0 is equivalent to a NODROP statement and is the default.

NODROP Statement

The NODROP statement is the default statement of the DROP statement, and is equivalent to DROP 0.

```
►►──NODROP─────────────────────────────────────────────────────►◄
```

The NODROP statement has no operands.

SEED Statement

The SEED statement defines the seed for the pseudo-random number generator used by the DROP statement debugging function. If the DROP statement is not zero and the SEED statement is not specified, the TCPIP virtual machine generates a seed based on the low-order bits of the hardware clock.

```
►►──SEED──random_number_seed──────────────────────────────────────►◄
```

Operands

random_number_seed	Specifies an integer in the range of -2 147 483 648 to 2 147 483 647.

Chapter 31. Using Source Code Libraries

At installation, source code is loaded to 3TCPIPA0 2B3, and object code is loaded to 3TCPIPA0 2B2.

Much of TCP/IP is coded in the Pascal language. Several components are written in the C language or System/370 assembler. The following convention is used for file types:

Code	**Description**
ASSEMBLE	Assembly language program files
C	C language program files
COPY	Pascal language include files or assembler DSECTS
H	C language include files
MACRO	Assembler macro files
PASCAL	Pascal language program files
CSQL, SQC	C program files with SQL preprocessor statements.

To write application programs that interface with TCP/IP, use the following EXECs, as appropriate:

- VMFASM EXEC, VMFHASM EXEC, or VMFHLASM EXEC
- VMFPAS EXEC
- VMFC EXEC
- TCPTXT EXEC
- TCPLOAD EXEC
- TCPCOMP EXEC.

Note: The VMFASM, VMFHASM, and VMFHLASM execs are part of the VMSES/E component of z/VM. The other execs are part of TCP/IP.

These EXECs are explained in the following sections.

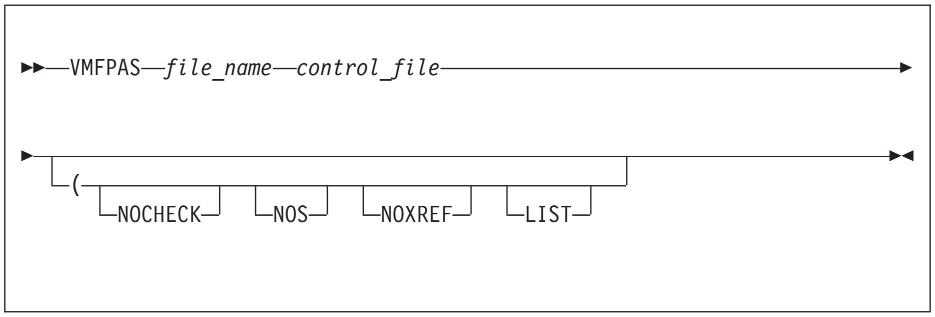
VMFASM EXEC, VMFHASM EXEC, and VMFHLASM EXEC

Use the VMFASM EXEC, VMFHASM EXEC, or VMFHLASM EXEC to update and compile assembler source code. For more information about these commands, see the *IBM VMSES/E Introduction and Reference*.

VMFPAS EXEC

The VMFPAS EXEC is similar in purpose to the VMSES/E VMFASM EXEC, but uses the CMS UPDATE command to apply updates to Pascal source and then compiles that source using the VS Pascal compiler.

Note: VMFPAS does not exploit the VMSES/E service environment. For example, VMSES/E Software Inventory Files are not used during VMFPAS processing, and the object files produced by VMFPAS do not comply with VMSES/E naming conventions.



Operands

file_name
: Specifies the file name of the Pascal source program. The file type must be PASCAL.

control_file
: Specifies the name of a control file that is used to apply the updates. The file type must be CNTRL.

NOCHECK
: Causes compilation without Pascal runtime checking. You should use this command to achieve better performance in fully debugged code.

NOS
: Does not produce a source listing.

NOXREF
: Does not produce a cross-reference.

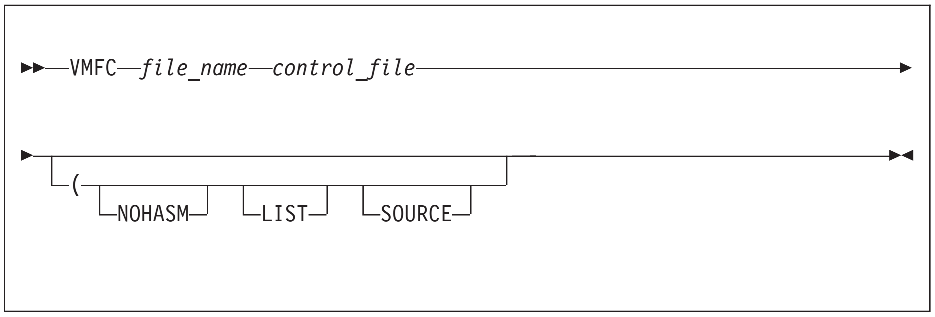
LIST
: Produces an assembler listing.

Note: The Pascal programs distributed with TCP/IP have been compiled with the VS Pascal Compiler and Library V1.2 (5668-767). You must use this compiler to ensure successful compilation and compatibility with the distributed code.

VMFC EXEC

The VMFC EXEC is similar in purpose to the VMSES/E VMFASM EXEC, but uses the CMS UPDATE command to apply updates to C source and then compiles that source using the CC exec provided with the IBM C for VM/ESA compiler.

Note: VMFC does not exploit the VMSES/E service environment. For example, VMSES/E Software Inventory Files are not used during VMFPAS processing, and the object files produced by VMFPAS do not comply with VMSES/E naming conventions.



Operands

file_name	Specifies the file name of the C source program. The file type must be C.
control_file	Specifies the file name of a control file that is used to apply the updates. The file type must be CNTRL.
NOHASM	Invokes assembler-XF. The default invokes the assembler specified at installation time.
LIST	Produces an assembler listing.
SOURCE	Produces a source listing.

Note: The C programs distributed with TCP/IP have been compiled with the IBM C for VM/ESA Compiler, Version 3 Release 1 (5654-033) and the IBM Language Environment for MVS & VM, Version 1 Release 8 (5688-198) products.

TCPTXT EXEC

The TCPTXT EXEC constructs a text library (TXTLIB) when it is given a list of text file names and a control file.

Almost all TXTLIBs used by TCP/IP are built using VMSES/E. If you have to build one that is not, you can use the TCPTXT command. Before running the command, make sure the latest serviced level of each text deck included in the text library is available, with the appropriate file type.

Use TCPTXT to construct the XMLIB TXTLIB.

```
►►──TCPTXT──load_list──control_file──────────────────────────────►◄
```

Operands

load_list	Specifies a file name with file type LOADLIST that contains the text file names to be included in the TXTLIB.
control_file	Specifies the file name of a control file that is used to select the text files. The file type must be CNTRL.

TCPLOAD EXEC

When running TCPLOAD, you must access all disks containing object files as extensions of the A disk. The TCPLOAD EXEC generates a module when given a list of text file names and a control file.

Almost all MODULEs used by TCP/IP are built using VMSES/E. If you have to build one that is not, you can use the TCPLOAD command. Before running the command, make sure the latest serviced level of each text deck included in the text library is available, with the appropriate file type.

```
►►──TCPLOAD──load_list──control_file──type─────────────────────────────────────►

►──┬──────────────────────────────────────────┬─────────────────────────────►◄
   └─(──┬────┬──┬─────────────────────────┬──┘
        └─XA─┘  │           ┌──────────┐  │
                └─TXTLIB────▼─filename─┴──┘
```

Operands

load_list
: Specifies a file name with a file type of LOADLIST that contains text file names to be included in the load module. The first line in the *load_list* specifies the name of the main object module. Subsequent lines specify additional object modules to be included in the load module.

control_file
: Specifies the file name of the control file that is used to select the text files. The file type must be CNTRL.

type
: Specifies the programming language.

  C Includes SCEELKED, CMSLIB, TCPASCAL, and TCPLANG TXTLIBs.

  C-ONLY
  Includes SCEELKED and CMSLIB TXTLIBs.

  PASCAL
  Includes TCPASCAL and TCPLANG TXTLIBs.

  Note: The COMMTXT TXTLIB is always included in the GLOBAL TXTLIB statement.

XA
: Includes the options AMODE 31 RMODE ANY on the LOAD and GENMOD commands. It also includes the TCPXXA TXTLIB.

TXTLIB *filename*
: Allows you to specify up to 50 TXTLIBs that are added to the GLOBAL TXTLIB command. RPC users should specify TXTLIB RPCLIB.

Note: CMSLIB is only needed when running in 370 mode.

The generated module has the same file name as the file name of the load list. The load lists shipped with TCP/IP are:

```
MODULE                  DISK

SAMPLE_C LOADLIST       TCPMAINT 592
SAMPLE_S LOADLIST       TCPMAINT 592
```

Note: You can ignore the following message from the preloading step:

```
DMSPRE236E UNRESOLVED EXTERNAL REFERENCE(S) ENCOUNTERED
```

TCPCOMP EXEC

The TCPCOMP EXEC recompiles or assembles all the files listed in a LOADLIST file. This command is useful when you need to recompile all the files of a component.

```
►►──TCPCOMP──load_list──control_file──────────────────────────────────►◄
```

Operands

load_list	Specifies a file that has a file type of LOADLIST or LKEDCTRL that contains the file names to be recompiled or assembled.
control_file	Specifies the file name of a control file that is used to apply the updates. The file type must be CNTRL.

Special Considerations

The following are special considerations that apply to rebuilding the TCP/IP code:

- To reassemble CMMALLOC and CMQSTOR, use either the H Assembler or HL Assembler.
- To reassemble NSDBSQL ASMSQL, link to the SQL system minidisk and execute the SQLPP exec.

Appendix A. Using TCP/IP with an External Security Manager

Some of the TCP/IP servers are involved in making system resources available to clients. These servers ensure that the clients' credentials (user ID and password) are valid, and they ensure that the client accesses only those resources permitted by those credentials.

Credential validation and resource access are under the control of CP or an external security manager (ESM). IBM's Resource Access Control Facility (RACF) is an example of an external security manager that offers effective user authentication, resource access controls, and logging capabilities.

The following servers can be configured to interface with an ESM to provide system resource protection, if desired:

FTP	(FTPSERVE)
LP	(LPSERVE)
NDB	(NDBSRV*nn*)
NFS	(VMNFS)
REXEC	(REXECD)

This appendix describes the interfaces used by TCP/IP servers to access z/VM security-related services, and the specific actions that must be taken if you use RACF.

If you use an external security manager other than RACF, consult the appropriate publications for your security manager for similar configuration information.

Security Interfaces

All password validation, "Logon By" permission verification, and minidisk access control services are obtained through three CSL routines:

DMSESM	Enables access to external security manager services. This routine provides information that is used with DMSLINK and DMSPWCHK.
DMSPWCHK	Provides password and "Logon By" verification.
DMSLINK	Provides minidisk and virtual reader access services.

These CSL routines in turn access ESM services using the following interfaces:

RPIVAL	A command that determines if a VM user ID and password are valid.
RACROUTE	A macro that is used to verify "Logon By" privileges and to determine the permitted level of access to minidisks and virtual readers.

The *z/VM: CMS Callable Services Reference* contains detailed information on the operation of these CSL routines, including their interactions with the ESM.

These interfaces provide two levels at which your can implement your own security scheme:

- You can replace on or more of the CSL routines with those of your own creation, or those provided by another vendor, or
- You can provide your own password validation program to perform the function of RPIVAL, and a RACROUTE request handler.

In the descriptions that follow, it is assumed that the CSL routines provided by IBM are being used and that the default tag values in the IBM DTCPARMS file have not been overridden or changed.

Server Initialization

When :ESM_Enable.YES is specified in the DTCPARMS file for any of the servers listed on page 617, two things will happen when the server is started:

1. The RACROUTE macro request handler is enabled using the command identified on the :ESM_Racroute. tag in the DTCPARMS file (RPIUCMS or RPIDUMY, by default).

   The operation of RPIUCMS is defined in the *RACF External Security Interface (RACROUTE) Macro Reference for MVS and VM*. RPIDUMY is included with TCP/IP and is used in cases where a server requires password validation services, but does not use RACROUTE.
2. The servers will call DMSESM to obtain a security token that may be used to uniquely identify the server to the ESM.

The security token is included on calls to DMSPWCHK and DMSLINK. These routines use this token to determine whether their services should be provided using native CP functions or those of the ESM.

Client Authentication

Whenever a client requests access to the system, the server will call DMSPWCHK to ensure that the user ID and password provided are valid. Expired passwords may not be used to access the system and the servers do not provide a mechanism for clients to change the password.

If the “Logon By” form of FTP or NFS client login is used, both user IDs are passed to DMSPWCHK, which will ensure that the client’s own user ID and password are valid, and that the user has permission to “logon by” to the specified user ID (called the *target* ID).

Note: If your ESM does not provide a RACROUTE interface, **LOGONBY** user directory statements will be used to determine “Logon By” privileges.

If DMSESM returned a non-zero security token, DMSPWCHK will verify the client’s user ID and password using the command identified on the :ESM_Validate. tag in the DTCPARMS file. The default command is RPIVAL.

Resource Access

Whenever an FTP or NFS client requests access to a minidisk or virtual reader, the server calls DMSLINK. This routine determines if the client may access the requested resource.

For minidisks, DMSLINK also obtains a link to a minidisk. Minidisk passwords may or may not be required, depending on your external security manager configuration. If a password is required, but one was not provided, DMSLINK will indicate this condition and the servers will act accordingly. In the case of FTP, the client is prompted to issue the ACCT subcommand to provide a minidisk password. For NFS, the mount request is rejected, requiring the client to provide the password using the **MDISKPW=** parameter on the mount string.

Note: Incorrect passwords provided by FTP or NFS clients are not tracked or journaled by CP.

For a virtual reader, DMSLINK is called to determine whether the client may view and manipulate the contents of the reader queue (DMSLINK RC=0), or only send files to the queue (DMSLINK RC=4).

The DTCPARMS File

Three DTCPARMS file entries relate to TCP/IP's use of external security manager interfaces:

:ESM_Enable.
Indicates whether an External Security Manager (ESM) is to be used to authenticate and authorize access to resources managed by this server. When no value is specified for this tag the default is **NO**, in which case native CP security services (diagnose X'88') will be used.

:ESM_Validate.
Identifies a program to validate user IDs and passwords supplied by clients. When no value is specified for this tag the default is **RPIVAL**.

If a different program is used, it must follow the programming conventions (parameter format and return codes) used by RPIVAL. More information on the RPIVAL command may be found in *RACF Macros and Interfaces*.

:ESM_Racroute.
Identifies a program to initialize and terminate the RACROUTE environment. When no value is specified for this tag the default is **RPIUCMS**, a program provided by the RACF product.

:ESM_Racroute.`RPIDUMY` should be specified whenever you wish to use ESM password validation, but want to provide native CP minidisk protection. RPIDUMY provides a non-zero security token on the DMSESM call, but will defer to CP whenever a RACROUTE authorization call is made.

The default values for the :ESM_Validate. and :ESM_Racroute. tags are appropriate for RACF. If an ESM other than RACF is in use, it may be necessary to define alternate values for the :ESM_Validate. and :ESM_Racroute. tags.

These tags may be specified on the `:Type.Server` entry for a server, or they may be specified on the appropriate `:Type.Class` entry in the DTCPARMS file. Providing this information at the class level ensures that all servers of the same class will use the specified ESM services.

If you choose to modify the class entries, copy them from IBM DTCPARMS to a local DTCPARMS file — do not modify IBM DTCPARMS because it may be replaced by the application of service or the upgrade to a new release of TCP/IP. With each service application or upgrade, you will need to verify that any changes introduced by IBM are incorporated into your local DTCPARMS file.

The suggested alternative is to use the global server profile exit (TCPRUNXT EXEC) to set these tags. By using the profile exit, you may make decisions based on server user ID or server class. See "Chapter 3. General TCP/IP Server Configuration" on page 17 for more information on using profile exits.

For more information about using these tags, see "DTCPARMS Tags" on page 21.

Minidisk Security

The FTP and NFS servers link to minidisks on behalf of clients. They do not have the same function as a traditional "batch" machine and, therefore, should not have the same minidisk security policy as a batch machine.

To access minidisks, the FTP and NFS servers use diagnose X'D4' to change their identity. It is important that your ESM implement a security policy for these servers that makes minidisk access decisions based solely on this *alternate user ID*, not the user ID of the server virtual machine (FTPSERVE, for example).

Failure to implement this security policy could result in unintended or undesirable access by a client to the FTP or NFS server's 191 disks or to TCP/IP configuration files which may contain sensitive data.

Using TCP/IP with RACF

The following procedures enable the server machines listed on page 617, to use the authorization interfaces provided by RACF. If you use a security manager product other than RACF, consult the appropriate product publications.

Before you begin, it will simplify TCP/IP server administration if you create one or more RACF groups that contain some or all of the TCP/IP servers that are installed. By doing so, you can grant a new server all necessary permissions by simply connecting it to the appropriate group(s).

1. Modify RACF installation exit **ICHRCX02** to **not** allow an alternate user to access resources that can be accessed by the FTP and NFS servers. See the *RACF System Programmer's Guide*, SC28-1343. Also see the previous section "Minidisk Security".
2. Give each FTP, NFS, REXEC, NDB, and LP server permission to perform password validation using the RPIVAL command (which uses diagnose X'A0', subcode 4):

   PERMIT DIAG0A0.VALIDATE CLASS(VMCMD) ID(*userid* | *groupid*) ACCESS(READ)
3. Enable the RACROUTE facility and permit all FTP and NFS servers to use it:
   a. Log on to a user ID that has the system-SPECIAL attribute and create a profile named ICHCONN in the FACILITY class:

      RDEFINE FACILITY ICHCONN UACC(NONE)
   b. Give UPDATE access authority to each server:

      PERMIT ICHCONN CLASS(FACILITY) ID(*userid* | *groupid*) ACCESS(UPDATE)
   c. Activate the FACILITY class if it is not already active:

      SETROPTS CLASSACT(FACILITY)
   d. Enable IUCV communications with the RACF service machine(s). If the CP directory entries of the FTP or NFS servers do not contain an IUCV ANY statement, add an IUCV statement that identifies the RACF service machines with which these servers will communicate. For example,

      IUCV RACFVM PRIORITY MSGLIMIT 1000

This is not required if the CP directory entries of all RACF service machines contain an IUCV ALLOW statement.

See the *RACF Security Administrator's Guide* for more information.

4. Permit the NFS and FTP servers to act on behalf of clients.
   a. Use the NFSPERM and/or FTPPERM commands for users with discrete VMBATCH profiles.
   b. Give the FTP and NFS servers CONTROL access to a generic VMBATCH profile so users without a discrete VMBATCH profile may use FTP and NFS services:

      PERMIT ** CLASS(VMBATCH) ID(*userid* | *groupid***) ACCESS(UPDATE)**

   Both techniques can be used if some users have discrete profiles and others do not. See "FTPPERM and NFSPERM" for additional information.

   See *z/VM: Planning and Administration* for additional information on the IUCV statement.
5. Modify the DTCPARMS server entries for all servers listed on page 617, to contain:
   - `:ESM_Enable.YES`
   - `:VMLINK.RACFVM 305` or some other minidisk, as required, to provide access to the RPIVAL MODULE.

   You may find it more convenient to make these changes on a class basis, instead of an individual server basis, to ensure that all servers of the same class will have the same security characteristics.

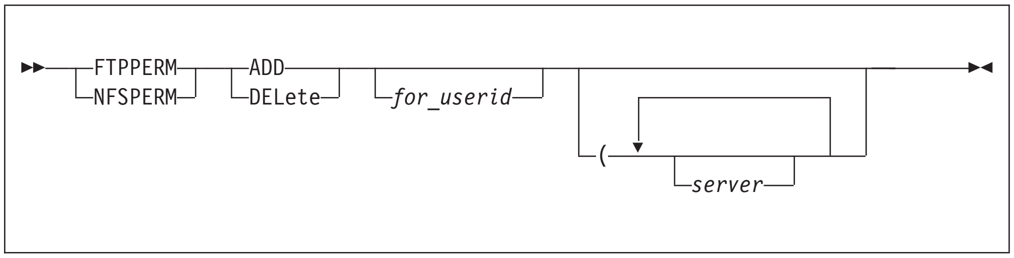
   The global profile exit, TCPRUNXT EXEC, provides an ideal way to set these tags. See "Chapter 3. General TCP/IP Server Configuration" on page 17 for more information on using profile exits.
6. Give the TCP/IP service user ID, P735FALT, UPDATE permission to all server A-disks as well as TCPMAINT 591, 592, and 198.

FTPPERM and NFSPERM

The FTPPERM and NFSPERM commands are used to allow the FTP and NFS servers to access all of the disks that a user would have access to if that user were logged on. This is accomplished by granting the servers CONTROL access to the user's VMBATCH profile.

If a user does not have a discrete VMBATCH profiles, FTPPERM and NFSPERM will issue a warning message. In this case, the FTP and NFS servers must instead be given CONTROL access to a generic VMBATCH profile.

Where discrete profiles exist, FTPPERM and NFSPERM must be run by each user who wishes to use FTP or NFS. Alternatively, the system administrator may issue these commands on the users' behalf.



Parameter	Description
ADD	Permits the FTP or NFS servers to act on the user’s behalf when accessing minidisks.
DELete	Removes permission for the FTP or NFS servers to act on the user’s behalf when accessing minidisks.
for_userid	Is the VM user ID on whose behalf this command is being issued. This parameter is used only by the system administrator.
server	The list of FTP or NFS server virtual machines that are affected. If no list is provided, the default is obtained from the **FTP_SERVERS** or **NFS_SERVERS** GLOBALV variable in the **TCPIP** group. If no GLOBALV variable has been set, the default is FTPSERVE for FTPPERM, and VMNFS for NFSPERM. It may be useful to set these GLOBALV variables in the SYSPROF EXEC so that users do not need to know the names of all of the servers that are present in your environment.

Examples

The following example gives FTPSERVE and FTPSERV2 permission to act on user ALAN’s behalf when accessing minidisks.

```
ftpperm add alan (ftpserve ftpserv2
Ready;
```

Because NFSPERM and FTPPERM perform the same function, you can use one command to give permission to FTP and NFS servers at the same time. This following example gives both FTPSERVE and VMNFS the needed permissions.

```
globalv select tcpip setl ftp_servers FTPSERVE VMNFS
Ready;

ftpperm add
Ready;
```

Appendix B. Related Protocol Specifications

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

Many features of TCP/IP for VM are based on the following RFCs:

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

RFCs

RFC	Title	Author
1033	*Domain Administrators Operations Guide*	M. Lottor
1034	*Domain Names—Concepts and Facilities*	P.V. Mockapetris
1035	*Domain Names—Implementation and Specification*	P.V. Mockapetris
1042	*Standard for the Transmission of IP Datagrams over IEEE 802 Networks*	J.B. Postel, J.K. Reynolds
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
1091	*Telnet Terminal-Type Option*	J. VanBokkelen
1094	*NFS: Network File System Protocol Specification*	Sun Microsystems Incorporated
1112	*Host Extensions for IP Multicasting*	S. Deering
1118	*Hitchhikers Guide to the Internet*	E. Krol
1122	*Requirements for Internet Hosts-Communication Layers*	R.T. Braden
1123	*Requirements for Internet Hosts-Application and Support*	R.T. Braden
1155	*Structure and Identification of Management Information for TCP/IP-Based Internets*	M.T. Rose, K. McCloghrie
1156	*Management Information Base for Network Management of TCP/IP-based Internets*	K. McCloghrie, M.T. Rose
1157	*Simple Network Management Protocol (SNMP),*	J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin
1179	*Line Printer Daemon Protocol*	The Wollongong Group, L. McLaughlin III
1180	*TCP/IP Tutorial,*	T. J. Socolofsky, C.J. Kale
1183	*New DNS RR Definitions* (Updates RFC 1034, RFC 1035)	C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris,
1187	*Bulk Table Retrieval with the SNMP*	M.T. Rose, K. McCloghrie, J.R. Davin
1188	*Proposed Standard for the Transmission of IP Datagrams over FDDI Networks*	D. Katz
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

RFC	Title	Author
1267	*A Border Gateway Protocol 3 (BGP-3)*	K. Lougheed, Y. Rekhter
1268	*Application of the Border Gateway Protocol in the Internet*	Y. Rekhter, P. Gross
1269	*Definitions of Managed Objects for the Border Gateway Protocol (Version 3)*	S. Willis, J. Burruss
1293	*Inverse Address Resolution Protocol*	T. Bradley, C. Brown
1270	*SNMP Communications Services*	F. Kastenholz, ed.
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
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
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
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
1440	*SIFT/UFT:Sender-Initiated/Unsolicited File Transfer*	R. Troth
1483	*Multiprotocol Encapsulation over ATM Adaptation Layer 5*	J. Heinanen
1540	*IAB Official Protocol Standards*	J.B. Postel
1583	*OSPF Version 2*	J.Moy
1647	*TN3270 Enhancements*	B. Kelly
1700	*Assigned Numbers*	J.K. Reynolds, J.B. Postel
1723	*RIP Version 2 — Carrying Additional Information*	G. Malkin
1813	*NFS Version 3 Protocol Specification*	B. Callaghan, B. Pawlowski, P. Stauback, Sun Microsystems Incorporated
2225	*Classical IP and ARP over ATM*	M. Laubach, J. Halpern

These documents can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Many RFCs are available online. Hard copies of all RFCs are available from the NIC, either individually or on a subscription basis. Online copies are available using FTP from the NIC at `nic.ddn.mil`. Use FTP to download the files, using the following format:

```
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS
```

Where:

nnnn	Is the RFC number.
TXT	Is the text format.
PS	Is the PostScript format.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC` *nnnn* for text versions or a subject line of `RFC` *nnnn*`.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of `RFC INDEX`.

For more information, contact `nic@nic.ddn.mil`. Information is also available through http://www.internic.net.

Appendix C. Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this book.

AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
API	Application Program Interface
APPC	Advanced Program-to-Program Communications
APPN®	Advanced Peer-to-Peer Networking®
ARP	Address Resolution Protocol
ASCII	American National Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
AUI	Attachment Unit Interface
BFS	Byte File System
BIOS	Basic Input/Output System
BNC	Bayonet Neill-Concelman
CCITT	Comite Consultatif International Telegraphique et Telephonique. The International Telegraph and Telephone Consultative Committee
CETI	Continuously Executing Transfer Interface
CLAW	Common Link Access to Workstation
CLIST	Command List
CMS	Conversational Monitor System
CP	Control Program
CPI	Common Programming Interface
CREN	Corporation for Research and Education Networking
CSD	Corrective Service Diskette
CTC	Channel-to-Channel
CU	Control Unit
CUA®	Common User Access®
DASD	Direct Access Storage Device
DBCS	Double Byte Character Set
DLL	Dynamic Link Library
DNS	Domain Name System
DOS	Disk Operating System
DPI	Distributed Program Interface
EBCDIC	Extended Binary-Coded Decimal Interchange Code
ELANS	IBM Ethernet LAN Subsystem
EISA	Enhanced Industry Standard Adapter
ESCON®	Enterprise Systems Connection Architecture®
FAT	File Allocation Table
FDDI	Fiber Distributed Data Interface
FTAM	File Transfer Access Management
FTP	File Transfer Protocol
FTP API	File Transfer Protocol Applications Programming Interface
GCS	Group Control System
GDDM®	Graphical Data Display Manager
GDDMXD	Graphics Data Display Manager Interface for X Window System
GDF	Graphics Data File
HCH	HYPERchannel device

Abbreviations and Acronyms

HIPPI	High Performance Parallel Interface
HPFS	High Performance File System
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
ILANS	IBM Token-Ring LAN Subsystem
IP	Internet Protocol
IPL	Initial Program Load
ISA	Industry Standard Adapter
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IUCV	Inter-User Communication Vehicle
JES	Job Entry Subsystem
JIS	Japanese Institute of Standards
JCL	Job Control Language
LAN	Local Area Network
LAPS	LAN Adapter Protocol Support
LCS	IBM LAN Channel Station
LPD	Line Printer Daemon
LPQ	Line Printer Query
LPR	Line Printer Client
LPRM	Line Printer Remove
LPRMON	Line Printer Monitor
LU	Logical Unit
MAC	Media Access Control
Mbps	Megabits per second
MBps	Megabytes per second
MCA	Micro Channel® Adapter
MIB	Management Information Base
MIH	Missing Interrupt Handler
MILNET	Military Network
MHS	Message Handling System
MTU	Maximum Transmission Unit
MVS	Multiple Virtual Storage
MX	Mail Exchange
NCP	Network Control Program
NDIS	Network Driver Interface Specification
NFS	Network File System
NIC	Network Information Center
NLS	National Language Support
NSFNET	National Science Foundation Network
OS/2®	Operating System/2®
OSA	Open Systems Adapter
OSF	Open Software Foundation, Inc.
OSI	Open Systems Interconnection
OSIMF/6000	Open Systems Interconnection Messaging and Filing/6000
OV/MVS	OfficeVision/MVS™
OV/VM	OfficeVision/VM™
PAD	Packet Assembly/Disassembly
PC	Personal Computer
PCA	Parallel Channel Adapter
PDN	Public Data Network
PDU	Protocol Data Units
PING	Packet Internet Groper

PIOAM	Parallel I/O Access Method
POP	Post Office Protocol
PROFS®	Professional Office Systems
PSCA	Personal System Channel Attach
PSDN	Packet Switching Data Network
PU	Physical Unit
PVM	Passthrough Virtual Machine
RACF	Resource Access Control Facility
RARP	Reverse Address Resolution Protocol
REXEC	Remote Execution
REXX	Restructured Extended Executor Language
RFC	Request For Comments
RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computer
RPC	Remote Procedure Call
RSCS	Remote Spooling Communications Subsystem
SAA	System Application Architecture
SBCS	Single Byte Character Set
SDLC	Synchronous Data Link Control
SFS	Shared File System
SLIP	Serial Line Internet Protocol
SMIL	Structure for Management Information
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
SOA	Start of Authority
SPOOL	Simultaneous Peripheral Operations Online
SQL	IBM Structured Query Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TFTP	Trivial File Transfer Protocol
TSO	Time Sharing Option
TTL	Time-to-Live
UDP	User Datagram Protocol
VGA	Video Graphic Array
VM	Virtual Machine
VMCF	Virtual Machine Communication Facility
VM/ESA	Virtual Machine/Enterprise System Architecture
VMSES/E	Virtual Machine Serviceability Enhancements Staged/Extended
VTAM®	Virtual Telecommunications Access Method
WAN	Wide Area Network
XDR	eXternal Data Representation

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10594-1785
U.S.A.

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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.

Programming Interface Information

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

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

PI

<....Programming Interface information....>

PI end

Trademarks

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

ACF/VTAM	AIX
Advanced Peer-to-Peer Networking	APL2
APPN	BookManager
Common User Access	CUA
DB2	DFSMS/VM
Enterprise Systems Connection Architecture	
GDDM	IBM
IBMLink	IPDS
Language Environment	Micro Channel
MVS	Network Station
OfficeVision	OfficeVision/MVS
OfficeVision/VM	OpenEdition
OpenExtensions	Operating System/2
OS/2	Presentation Manager
OS/390	PROFS
RACF	RISC System/6000
RS/6000	S/390
SP	SQL/DS
System/370	VM/ESA
VTAM	z/VM

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

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

Glossary

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

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

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

Numerics

3172. IBM Interconnect Controller.

3174. IBM Establishment Controller.

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

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

6611. IBM Network Processor.

8232. IBM LAN Station.

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

A

abend. The abnormal termination of a program or task.

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

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

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

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

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

address. The unique code assigned to each device or workstation connected to a network. A standard internet address uses a two-part, 32-bit address field. The first part of the address field contains the network address; the second part contains the local address.

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

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

Address Resolution Protocol (ARP). A protocol used to dynamically bind an internet address to a hardware address. ARP is implemented on a single physical network and is limited to networks that support broadcast addressing.

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

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

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

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

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

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

AIX. Advanced Interactive Executive.

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

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

ANSI. American National Standards Institute.

API. Application Program Interface.

APPC. Advanced Program-to-Program Communications.

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

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

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

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

ARP. Address Resolution Protocol.

ARPA. Advanced Research Projects Agency.

ARPANET. Advanced Research Projects Agency Network.

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

ASN.1. Abstract Syntax Notation One.

ASYNC. Asynchronous.

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

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

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

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

AUI. Attachment Unit Interface.

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

authentication server. The service that reads a Kerberos database to verify that a client making a request for access to an end-service is the client named in the request. The authentication server provides an authenticated client ticket as permission to access the ticket-granting server.

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

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

B

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

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

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

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

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

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

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

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

BER. Basic Encoding Rules.

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

BFS. Byte File System.

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

BIOS. Basic Input/Output System.

BITNET. Because It's Time NETwork.

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

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

BNC. Bayonet Neill-Concelman.

BOOTPD. Bootstrap Protocol Daemon.

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

bridge. A router that connects two or more networks and forwards packets among them. The operations carried out by a bridge are done at the physical layer and are transparent to TCP/IP and TCP/IP routing. A functional unit that connects two local area networks (LANs) that use the same logical link control (LLC) procedures but may use different medium access control (MAC) procedures.

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

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

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

BSD. Berkeley Software Distribution.

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

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

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

C

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

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

CCITT. Comite Consultatif International Telegraphique et Telephonique.

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

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

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

CICS. Customer Information Control System.

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

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

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

CLAW. Common Link Access to Workstation.

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

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

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

CLIST. Command List.

CLPA. Create Link Pack Area.

CMS. Conversational Monitor System.

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

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

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

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

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

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

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

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

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

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

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

Control Program (CP). The VM operating system that manages the real processor's resources and is responsible for simulating System/370s or 390s for individual users.

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

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

CP. Control Program.

Create Link Pack Area (CLPA). A parameter specified at startup, which says to create the link pack area.

CREN. Corporation for Research and Educational Networking.

CSMA/CD. Carrier Sense Multiple Access with Collision Detection.

CSNET. Computer and Science Network.

Customer Information Control System (CICS). An IBM-licensed program that enables transactions entered at remote terminals to be processed concurrently by user written application programs. It includes facilities for building, using, and maintaining databases.

D

daemon. A background process usually started at system initialization that runs continuously and performs a function required by other processes. Some daemons are triggered automatically to perform their task; others operate periodically.

DASD. Direct Access Storage Device.

DARPA. Defense Advanced Research Projects Agency.

DATABASE 2 (DB2). An IBM relational database management system for the MVS operating system.

database administrator (DBA). An individual or group responsible for the rules by which data is accessed and stored. The DBA is usually responsible for database integrity, security, performance and recovery.

datagram. A basic unit of information that is passed across the internet, it consists of one or more data packets.

data link layer. Layer 2 of the OSI (Open Systems Interconnection) model; it defines protocols governing data packetizing and transmission into and out of each node.

data set. The major unit of data storage and retrieval in MVS, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. Synonymous with *file* in VM and OS/2.

DB2. DATABASE 2.

DBA. Database administrator.

DBCS. Double Byte Character Set.

DDN. Defense Data Network.

decryption. The unscrambling of data using an algorithm that works under the control of a key. The key allows data to be protected even when the algorithm is unknown. Data is unscrambled after transmission.

default. A value, attribute, or option that is assumed when none is explicitly specified.

Defense Advanced Research Projects Agency (DARPA). The U.S. government agency that funded the ARPANET.

Defense Data Network (DDN). Comprises the MILNET and several other Department of Defense networks.

destination node. The node to which a request or data is sent.

DHCPD. Dynamic Host Configuration Protocol Daemon.

Direct Access Storage Device (DASD). A device in which access to data is independent of where data resides on the device.

directory. A named grouping of files in a file system.

Disk Operating System (DOS). An operating system for computer systems that use disks and diskettes for auxiliary storage of programs and data.

display terminal. An input/output unit by which a user communicates with a data-processing system or sub-system. Usually includes a keyboard and always provides a visual presentation of data; for example, an IBM 3179 display.

Distributed Program Interface (DPI). A programming interface that provides an extension to the function provided by the SNMP agents.

DLL. Dynamic Link Library.

DNS. Domain Name System.

domain. In an internet, a part of the naming hierarchy. Syntactically, a domain name consists of a sequence of names (labels) separated by periods (dots).

Domain Name System (DNS). A system in which a resolver queries name servers for resource records about a host.

domain naming. A hierarchical system for naming network resources.

DOS. Disk Operating System.

dotted-decimal notation. The syntactic representation for a 32-bit integer that consists of four 8-bit numbers, written in base 10 and separated by periods (dots). Many internet application programs accept dotted decimal notations in place of destination machine names.

double-byte character set (DBCS). A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS.

doubleword. A contiguous sequence of bits or characters that comprises two computer words and is capable of being addressed as a unit.

DPI. Distributed Program Interface.

Dynamic Host Configuration Protocol Daemon (DHCPD). The DHCP daemon (DHCPD server) responds to client requests for boot information using information contained in a DHCP machine file. This information includes the IP address of the client, the IP address of the TFTP daemon, and information about the files to request from the TFTP daemon.

dynamic resource allocation. An allocation technique in which the resources assigned for execution of computer programs are determined by criteria applied at the moment of need.

dynamic link library (DLL). A module containing dynamic link routines that is linked at load or run time.

E

EBCDIC. Extended binary-coded decimal interchange code.

EGP. Exterior Gateway Protocol.

encapsulation. A process used by layered protocols in which a lower-level protocol accepts a message from a higher-level protocol and places it in the data portion of the low-level frame. As an example, in Internet terminology, a packet would contain a header from the physical layer, followed by a header from the network layer (IP), followed by a header from the transport layer (TCP), followed by the application protocol data.

encryption. The scrambling or encoding of data using an algorithm that works under the control of a key. The key allows data to be protected even when the algorithm is unknown. Data is scrambled prior to transmission.

ES/9370 Integrated Adapters. An adapter you can use in TCP/IP for VM to connect into Token-Ring networks and Ethernet networks, as well as TCP/IP networks that support X.25 connections.

Ethernet. The name given to a local area packet-switched network technology invented in the early 1970s by Xerox**, Incorporated. Ethernet uses a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) mechanism to send packets.

EXEC. In a VM operating system, a user-written command file that contains CMS commands, other user-written commands, and execution control statements, such as branches.

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

extended character. A character other than a 7-bit ASCII character. An extended character can be a 1-bit code point with the 8th bit set (ordinal 128-255) or a 2-bit code point (ordinal 256 and greater).

Exterior Gateway Protocol (EGP). A reachability routing protocol used by gateways in a two-level internet.

eXternal Data Representation (XDR). A standard developed by Sun Microsystems, Incorporated for representing data in machine-independent format.

F

FAT. File Allocation Table.

FDDI. Fiber Distributed Data Interface. Also used to abbreviate Fiber Optic Distributed Data Interface.

Fiber Distributed Data Interface (FDDI). The ANSI standard for high-speed transmission over fiber optic cable.

Fiber Optic Network. A network based on the technology and standards that define data transmission using cables of glass or plastic fibers carrying visible light. Fiber optic network advantages are: higher transmission speeds, greater carrying capacity, and lighter, more compact cable.

file. In VM and OS/2, a named set of records stored or processed as a unit. Synonymous with *data set* in MVS.

File Allocation Table (FAT). A table used to allocate space on a disk for a file.

File Transfer Access and Management (FTAM). An application service element that enables user application processes to manage and access a file system, which may be distributed.

File Transfer Protocol (FTP). A TCP/IP protocol used for transferring files to and from foreign hosts. FTP also provides the capability to access directories. Password protection is provided as part of the protocol.

foreign host. Any machine on a network that can be interconnected.

foreign network. In an internet, any other network interconnected to the local network by one or more intermediate gateways or routers.

foreign node. See *foreign host*.

frame. The portion of a tape on a line perpendicular to the reference edge, on which binary characters can be written or read simultaneously.

FTAM. File Transfer Access and Management.

FTP. File Transfer Protocol.

fullword. A computer word. In System/370, 32 bits or 4 bytes.

G

gadget. A windowless graphical object that looks like its equivalent like-named widget but does not support the translations, actions, or pop-up widget children supplied by that widget.

gateway. A functional unit that interconnects a local data network with another network having different protocols. A host that connects a TCP/IP network to a non-TCP/IP network at the application layer. See also *router*.

gather and scatter data. Two related operations. During the gather operation, data is taken from multiple buffers and transmitted. In the scatter operation, data is received and stored in multiple buffers.

GC. Graphics Context.

GContext. See *Graphics Context*.

GCS. Group Control System.

GDDM. Graphical Data Display Manager.

GDDMXD. Graphical Data Display Manager interface for X Window System. A graphical interface that formats and displays alphanumeric, data, graphics, and images on workstation display devices that support the X Window System.

GDF. Graphics data file.

Graphical Display Data Manager (GDDM). A group of routines that allows pictures to be defined and displayed procedurally through function routines that correspond to graphic primitives.

Graphics Context (GC). The storage area for graphics output. Also known as *GC* and *GContext*. Used only with graphics that have the same root and depth as the graphics content.

Group Control System (GCS) . A component of VM/ESA, consisting of a shared segment that you can Initial Program Load (IPL) and run in a virtual machine. It provides simulated MVS services and unique supervisor services to help support a native SNA network.

H

handle. A temporary data representation that identifies a file.

halfword. A contiguous sequence of bits or characters that constitutes half a fullword and can be addressed as a unit.

HASP. Houston automatic spooling priority system.

HDLC. High-level Data Link Control.

header file. A file that contains constant declarations, type declarations, and variable declarations and assignments. Header files are supplied with all programming interfaces.

High-level Data Link Control (HDLC). An ISO protocol for X.25 international communication.

High Performance File System (HPFS). An OS/2 file management system that supports high-speed buffer storage, long file names, and extended attributes.

hop count. The number of gateways or routers through which a packet passes on its way to its destination.

host. A computer connected to a network, which provides an access method to that network. A host provides end-user services and can be a client, a server, or a client and server simultaneously.

Houston automatic spooling priority system (HASP). A computer program that provides supplementary job management, data management, and task management functions such as control of job flow, ordering of tasks, and spooling.

HPFS. High Performance File System.

HYPERchannel Adapter. A network interface used to connect a TCP/IP for VM or MVS host into an existing TCP/IP HYPERchannel network, or to connect TCP/IP hosts together to create a TCP/IP HYPERchannel network.

I

IAB. Internet Activities Board.

ICMP. Internet Control Message Protocol.

IEEE. Institute of Electrical and Electronic Engineers.

IETF. Internet Engineering Task Force.

IGMP. Internet Group Management Protocol (IGMP).

IGP. Interior Gateway Protocol.

include file. A file that contains preprocessor text, which is called by a program, using a standard programming call. Synonymous with *header file*.

IMS. Information Management System.

Information Management System (IMS). A database/data communication (DB/DC) system that can manage complex databases and networks.

initial program load (IPL). The initialization procedure that causes an operating system to commence operation.

instance. Indicates a label that is used to distinguish among the variations of the *principal name*. An instance allows for the possibility that the same client or service can exist in several forms that require distinct authentication.

Institute of Electrical and Electronic Engineers (IEEE). An electronics industry organization.

Integrated Services Digital Network (ISDN). A digital, end-to-end telecommunication network that supports multiple services including, but not limited to, voice and data.

interactive. Pertaining to a program or a system that alternately accepts input and then responds. An interactive system is conversational; that is, a continuous dialog exists between user and system. See *batch*.

Interior Gateway Protocol (IGP). The protocol used to exchange routing information between collaborating routers in the Internet. RIP is an example of an IGP.

Internet. The largest internet in the world consisting of large national backbone nets (such as MILNET, NSFNET, and CREN) and a myriad of regional and local campus networks all over the world. The Internet uses the Internet protocol suite. To be on the Internet, you must have IP connectivity (be able to TELNET to, or PING, other systems). Networks with only electronic mail connectivity are not actually classified as being on the Internet.

Internet Activities Board (IAB). The technical body that oversees the development of the Internet suite of protocols (commonly referred to as TCP/IP). It has two task forces (the IRTF and the IETF) each charged with investigating a particular area.

Internet address. A 32-bit address assigned to hosts using TCP/IP. An internet address consists of a network number and a local address. Internet addresses are represented in a dotted-decimal notation and are used to route packets through the network.

Internet Engineering Task Force (IETF). One of the task forces of the IAB. The IETF is responsible for solving short-term engineering needs of the Internet.

International Organization for Standardization (ISO). An organization of national standards bodies from various countries established to promote development of standards to facilitate international exchange of goods and services, and develop cooperation in intellectual, scientific, technological, and economic activity.

internet or internetwork. A collection of packet switching networks interconnected by gateways, routers, bridges, and hosts to function as a single, coordinated, virtual network.

internet address. The unique 32-bit address identifying each node in an internet. See also *address*.

Internet Control Message Protocol (ICMP). The part of the Internet Protocol layer that handles error messages and control messages.

Internet Group Management Protocol (IGMP). IGMP is used by IP hosts to report their host group memberships to multicast routers.

Internet Protocol (IP). The TCP/IP layer between the higher level host-to-host protocol and the local network protocols. IP uses local area network protocols to carry packets, in the form of datagrams, to the next gateway, router, or destination host.

interoperability. The capability of different hardware and software by different vendors to effectively communicate together.

Inter-user communication vehicle (IUCV). A VM facility for passing data between virtual machines and VM components.

intrinsics X-Toolkit. A set management mechanism that provides for constructing and interfacing between composite X Window widgets, their children, and other clients. Also, intrinsics provide the ability to organize a collection of widgets into an application.

IP. Internet Protocol.

IP datagram. The fundamental unit of information passed across the Internet. An IP datagram contains source and destination addresses along with data and a number of fields that define such things as the length of the datagram, the header checksum, and flags to say whether the datagram can be (or has been) fragmented.

IPL. Initial Program Load.

ISDN. Integrated Services Digital Network.

ISO. International Organization for Standardization.

IUCV. Inter-User Communication Vehicle.

J

JCL. Job Control Language.

JES. Job Entry Subsystem.

JIS. Japanese Institute of Standards.

Job Control Language (JCL). A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

Job Entry Subsystem (JES). An IBM System/370 licensed program that receives jobs into the system and processes all output data produced by the jobs.

JUNET. The Japanese Academic and Research Network that connects various UNIX operating systems.

K

Kanji. A graphic character set consisting of symbols used in Japanese ideographic alphabets. Each character is represented by 2 bytes.

katakana. A character set of symbols used on one of the two common Japanese phonetic alphabets, which is used primarily to write foreign words phonetically. See also *kanji*.

Kerberos. A system that provides authentication service to users in a network environment.

Kerberos Authentication System. An authentication mechanism used to check authorization at the user level.

L

LaMail. The client that communicates with the OS/2 Presentation Manager to manage mail on the network.

LAN. Local area network.

Line Printer Client (LPR). A client command that allows the local host to submit a file to be printed on a remote print server.

Line Printer Daemon (LPD). The remote printer server that allows other hosts to print on a printer local to your host.

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

local area network (LAN). A data network located on the user's premises in which serial transmission is used for direct data communication among data stations.

local host. In an internet, the computer to which a user's terminal is directly connected without using the internet.

local network. The portion of a network that is physically connected to the host without intermediate gateways or routers.

logical character delete symbol. A special editing symbol, usually the at (@) sign, which causes CP to delete it and the immediately preceding character from the input line. If many delete symbols are consecutively entered, the same number of preceding characters are deleted from the input line.

Logical Unit (LU). An entity addressable within an SNA-defined network. LUs are categorized by the types of communication they support.

LPD. Line Printer Daemon.

LPR. Line Printer Client.

LU. Logical Unit.

LU-LU session. In SNA, a session between two logical units (LUs). It provides communication between two end users, or between an end user and an LU services component.

LU type. In SNA, the classification of an LU-LU session in terms of the specific subset of SNA protocols and options supported by the logical units (LUs) for that session.

M

MAC. Media Access Control.

mail gateway. A machine that connects two or more electronic mail systems (often different mail systems on different networks) and transfers messages between them.

Management Information Base (MIB). A standard used to define SNMP objects, such as packet counts and routing tables, that are in a TCP/IP environment.

mapping. The process of relating internet addresses to physical addresses in the network.

mask. A pattern of characters used to control retention or elimination of portions of another pattern of characters. To use a pattern of characters to control retention or elimination of another pattern of characters. A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

Maximum Transmission Unit (MTU). The largest possible unit of data that can be sent on a given physical medium.

media access control (MAC). The method used by network adapters to determine which adapter has access to the physical network at a given time.

Message Handling System (MHS). The system of message user agents, message transfer agents, message stores, and access units that together provide OSI electronic mail.

MHS. Message Handling System.

MIB. Management Information Base.

microcode. A code, representing the instructions of an instruction set, which is implemented in a part of storage that is not program-addressable.

MILNET. Military Network.

Military Network (MILNET). Originally part of the ARPANET, MILNET was partitioned in 1984 to make it possible for military installations to have reliable network service, while the ARPANET continued to be used for research. See *DDN*.

minidisk. Logical divisions of a physical direct access storage device.

modem (modulator/demodulator). A device that converts digital data from a computer to an analog signal that can be transmitted on a telecommunication line, and converts the analog signal received to data for the computer.

Motif. see OSF/Motif.

mouse. An input device that is used to move a pointer on the screen and select items.

MPROUTE. Multi-Path Routing. Implements the OSPF protocol described in RFC 1583, 1058, and 1723.

MTU. Maximum Transmission Unit.

multicast. The simultaneous transmission of data packets to a group of selected nodes on a network or subnetwork.

multiconnection server. A server that is capable of accepting simultaneous, multiple connections.

Multiple Virtual Storage (MVS). Implies MVS/370, the MVS/XA product, and the MVS/ESA product.

multitasking. A mode of operation that provides for the concurrent performance execution of two or more tasks.

MVS. Multiple Virtual Storage.

N

name server. The server that stores resource records about hosts.

National Science Foundation (NSF). Sponsor of the NSFNET.

National Science Foundation Network (NSFNET). A collection of local, regional, and mid-level networks in the U.S. tied together by a high-speed backbone. NSFNET provides scientists access to a number of supercomputers across the country.

NCP. Network Control Program.

NDB. Network Database.

NDIS. Network Driver Interface Specification.

Netman. This device keyword specifies that this device is a 3172 LAN Channel Station that supports IBM Enterprise-Specific SNMP Management Information Base (MIB) variables for 3172. TCP/IP for VM supports SNMP GET and SNMP GETNEXT operations to request and retrieve 3172 Enterprise-Specific MIB variables. These requests are answered only by those 3172 devices with the NETMAN option in the PROFILE TCPIP file.

NetView. A system 390-based, IBM-licensed program used to monitor, manage, and diagnose the problems of a network.

network. An arrangement of nodes and connecting branches. Connections are made between data stations. Physical network refers to the hardware that makes up a network. Logical network refers to the abstract organization overlaid on one or more physical networks. An internet is an example of a logical network.

network adapter. A physical device, and its associated software, that enables a processor or controller to be connected to a network.

network administrator. The person responsible for the installation, management, control, and configuration of a network.

Network Control Program (NCP). An IBM-licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

network database (NDB). An IBM-licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. NDB allows interoperability among different database systems, and uses RPC protocol with a client/server type of relationship. NDB is used for data conversion, security, I/O buffer management, and transaction management.

Network Driver Interface Specification (NDIS). An industry-standard specification used by applications as an interface with network adapter device drivers.

network elements. As defined in the SNMP architecture, network elements are gateways, routers, and hosts that contain management agents responsible for performing the network management functions requested by the network management stations.

network file system (NFS). The NFS protocol, which was developed by Sun Microsystems, Incorporated, allows computers in a network to access each other's file systems. Once accessed, the file system appears to reside on the local host.

Network Information Center (NIC). Originally there was only one, located at SRI International and tasked to serve the ARPANET (and later DDN) community. Today, there are many NICs operated by local, regional, and national networks all over the world. Such centers provide user assistance, document service, training, and more.

Network Job Entry (NJE). In object distribution, an entry in the network job table that specifies the system action required for incoming network jobs sent by a particular user or group of users. Each entry is identified by the user ID of the originating user or group.

network layer. Layer 3 of the Open Systems Interconnection (OSI) model; it defines protocols governing data routing.

network management stations. As defined in the SNMP architecture, network management stations, or SNMP clients, execute management applications that monitor and control network elements.

NFS. Network file system.

NIC. Network Information Center.

NJE. Network Job Entry.

node. In a network, a point at which one or more functional units connect channels or data circuits. In a network topology, the point at an end of a branch.

nonblocking mode. If the execution of the program cannot continue until some event occurs, the operating system does not suspend the program until that event occurs. Instead, the operating system returns an error message to the program.

NPSI. X.25 NCP Packet Switching Interface.

NSF. National Science Foundation.

NSFNET. National Science Foundation Network.

O

octet. A byte composed of eight binary elements.

Offload host. Any device that is handling the TCP/IP processing for the MVS host where TCP/IP for MVS is installed. Currently, the only supported Offload host is the 3172-3.

Offload system. Represents both the MVS host where TCP/IP for MVS is installed and the Offload host that is handling the TCP/IP Offload processing.

open system. A system with specified standards and that therefore can be readily connected to other systems that comply with the same standards.

Open Systems Interconnection (OSI). The interconnection of open systems in accordance with specific ISO standards. The use of standardized procedures to enable the interconnection of data processing systems.

Operating System/2 (OS/2). Pertaining to the IBM licensed program that can be used as the operating system for personal computers. The OS/2 licensed program can perform multiple tasks at the same time.

OS/2. Operating System/2.

OSF/Motif. OSF/Motif is an X Window System toolkit defined by Open Software Foundation, Inc. (OSF), which enables the application programmer to include standard graphic elements that have a 3-D appearance. Performance of the graphic elements is increased with gadgets and windowless widgets.

OSI. Open Systems Interconnection.

OSPF. Open Shortest Path First. An Interior Gateway Protocol that distributes routing information within a single Autonomous System.

out-of-band data. Data that is placed in a secondary channel for transmission. Primary and secondary communication channels are created physically by modulation on a different frequency, or logically by specifying a different logical channel. A primary channel can have a greater capacity than a secondary one.

OV. OfficeVision.

P

packet. A sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole.

Packet Switching Data Network (PSDN). A network that uses packet switching as a means of transmitting data.

parameter. A variable that is given a constant value for a specified application.

parse. To analyze the operands entered with a command.

passive open. The state of a connection that is prepared to provide a service on demand. Contrast with *active open*.

Partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

PC. Personal computer.

PCA. Personal Channel Attach.

PC Network. A low-cost, broadband network that allows attached IBM personal computers, such as IBM 5150 Personal Computers, IBM Computer ATs, IBM PC/XTs, and IBM Portable Personal Computers to communicate and to share resources.

PDS. Partitioned data set.

PDN. Public Data Network.

PDU. Protocol data unit.

peer-to-peer. In network architecture, any functional unit that resides in the same layer as another entity.

Personal Channel Attach (PCA). see Personal System Channel Attach.

Personal Computer (PC). A microcomputer primarily intended for stand-alone use by an individual.

Personal System Channel Attach (PSCA). An adapter card to connect a micro-channel based personal computer (or processor) to a System/370 parallel channel.

physical layer. Layer 1 of the Open Systems Interconnection (OSI) model; it details protocols governing transmission media and signals.

physical unit (PU). In SNA, the component that manages and monitors the resources, such as attached links and adjacent link stations, associated with a node, as requested by an SSPC via an SSPC-PU session. An SSPC activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links.

PING. The command that sends an ICMP Echo Request packet to a host, gateway, or router with the expectation of receiving a reply.

PM. Presentation Manager.

PMANT. In OS/2, the 3270 client terminal emulation program that is invoked by the PMANT command.

polling. On a multipoint connection or a point-to-point connection, the process whereby data stations are invited one at a time to transmit. Interrogation of devices for such purposes as to avoid contention, to determine operational status, or to determine readiness to send or receive data.

POP. Post Office Protocol.

port. An endpoint for communication between devices, generally referring to a logical connection. A 16-bit number identifying a particular Transmission Control Protocol or User Datagram Protocol resource within a given TCP/IP node.

PORTMAP. Synonymous with *Portmapper*.

Portmapper. A program that maps client programs to the port numbers of server programs. Portmapper is used with Remote Procedure Call (RPC) programs.

Post Office Protocol (POP). A protocol used for exchanging network mail.

presentation layer. Layer 6 of the Open Systems Interconnections (OSI) model; it defines protocols governing data formats and conversions.

Presentation Manager (PM). A component of OS/2 that provides a complete graphics-based user interface, with pull-down windows, action bars, and layered menus.

principal name. Specifies the unique name of a user (client) or service.

PostScript. A standard that defines how text and graphics are presented on printers and display devices.

process. A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. Any operation or combination of operations on data. A function being performed or waiting to be performed. A program in operation; for example, a daemon is a system process that is always running on the system.

Professional Office Systems (PROFS). IBM's proprietary, integrated office management system used for sending, receiving, and filing electronic mail, and a variety of other office tasks. PROFS has been replaced by OfficeVision. See *OfficeVision*.

PROFS. Professional Office Systems.

protocol. A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. Protocols can determine low-level details of machine-to-machine interfaces, such as the order in which bits from a byte are sent; they can also determine high-level exchanges between application programs, such as file transfer.

Protocol data unit (PDU). A set of commands used by the SNMP agent to request management station data.

protocol suite. A set of protocols that cooperate to handle the transmission tasks for a data communication system.

PSCA. Personal System Channel Attach.

PSDN. Packet Switching Data Network.

PU. Physical unit.

Public Data Network (PDN). A network established and operated by a telecommunication administration or by a Recognized Private Operating Agency (RPOA) for the specific purpose of providing circuit-switched, packet-switched, and leased-circuit services to the public.

Q

QDIO. Queued Direct I/O.

queue. A line or list formed by items in a system waiting for service; for example, tasks to be performed or messages to be transmitted. To arrange in, or form, a queue.

R

RACF. Resource access control facility.

RARP. Reverse Address Resolution Protocol.

read-only access. An access mode associated with a virtual disk directory that lets a user read, but not write or update, any file on the disk directory.

read/write access. An access mode associated with a virtual disk directory that lets a user read and write any file on the disk directory (if write authorized).

realm. One of the three parts of a Kerberos name. The realm specifies the network address of the principal name or instance. This address must be expressed as a fully qualified domain name, not as a "dot numeric" internet address.

recursion. A process involving numerous steps, in which the output of each step is used for the successive step.

reduced instruction-set computer (RISC). A computer that uses a small, simplified set of frequently used instructions for rapid execution.

reentrant. The attribute of a program or routine that allows the same copy of a program or routine to be used concurrently by two or more tasks.

Remote Execution Protocol (REXEC). A protocol that allows the execution of a command or program on a foreign host. The local host receives the results of the command execution. This protocol uses the REXEC command.

remote host. A machine on a network that requires a physical link to interconnect with the network.

remote logon. The process by which a terminal user establishes a terminal session with a remote host.

Remote Procedure Call (RPC). A facility that a client uses to request the execution of a procedure call from a server. This facility includes a library of procedures and an eXternal data representation.

Remote Spooling Communications Subsystem (RSCS). An IBM-licensed program that transfers spool files, commands, and messages between VM users, remote stations, and remote and local batch systems, through HASP-compatible telecommunication facilities.

Request For Comments (RFC). A series of documents that covers a broad range of topics affecting internetwork communication. Some RFCs are established as internet standards.

resolver. A program or subroutine that obtains information from a name server or local table for use by the calling program.

resource access control facility (RACF). An IBM-licensed program that provides for access control by identifying and by verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

resource records. Individual records of data used by the Domain Name System. Examples of resource records include the following: a host's Internet Protocol addresses, preferred mail addresses, and aliases.

response unit (RU). In SNA, a message unit that acknowledges a request unit. It may contain prefix information received in a request unit. If positive, the response unit may contain additional information such as session parameters in response to BIND SESSION. If negative, it contains sense data defining the exception condition.

Restructured Extended Executor (REXX) language. A general purpose programming language, particularly suitable for EXEC procedures, XEDIT macros, or programs for personal computing. Procedures, XEDIT macros, and programs written in this language can be interpreted by the Procedures Language VM/REXX interpreter.

return code. A code used to influence the execution of succeeding instructions. A value returned to a program to indicate the results of an operation requested by that program.

Reverse Address Resolution Protocol (RARP). A protocol that maintains a database of mappings between physical hardware addresses and IP addresses.

REXEC. Remote Execution Protocol.

REXX. Restructured Extended Executor language.

RFC. Request For Comments.

RIP. Routing Information Protocol.

RISC. Reduced instruction-set computer.

router. A device that connects networks at the ISO Network Layer. A router is protocol-dependent and connects only networks operating the same protocol. Routers do more than transmit data; they also select the best transmission paths and optimum sizes for packets. In TCP/IP, routers operate at the Internetwork layer. See also *gateway*.

Routing Information Protocol (RIP). The protocol that maintains routing table entries for gateways, routers, and hosts.

routing table. A list of network numbers and the information needed to route packets to each.

RPC. Remote Procedure Call.

RSCS. Remote Spooling Communications Subsystem.

RU. Response unit.

S

SAA. Systems Application Architecture.

SBCS. Single Byte Character Set.

SDLC. Synchronous data link control.

Sendmail. The OS/2 mail server that uses Simple Mail Transfer Protocol to route mail from one host to another host on the network.

serial line. A network media that is a de facto standard, not an international standard, commonly used for point-to-point TCP/IP connections. Generally, a serial line consists of an RS-232 connection into a modem and over a telephone line.

semantics. The relationships of characters or groups of characters to their meanings, independent of the manner of their interpretation and use. The relationships between symbols and their meanings.

server. A function that provides services for users. A machine can run client and server processes at the same time.

SFS. Shared File System.

Shared File System (SFS). A part of CMS that lets users organize their files into groups known as *directories* and selectively share those files and directories with other users.

Simple Mail Transfer Protocol (SMTP). A TCP/IP application protocol used to transfer mail between users on different systems. SMTP specifies how mail systems interact and the format of control messages they use to transfer mail.

Simple Network Management Protocol (SNMP). A protocol that allows network management by elements, such as gateways, routers, and hosts. This protocol provides a means of communication between network elements regarding network resources.

simultaneous peripheral operations online (SPOOL). (Noun) An area of auxiliary storage defined to temporarily hold data during its transfer between peripheral equipment and the processor. (Verb) To use auxiliary storage as a buffer storage to reduce processing delays when transferring data between peripheral equipment and the processing storage of a computer.

single-byte character set (SBCS). A character set in which each character is represented by a one-byte code. Contrast with double-byte character set.

SMI. Structure for Management Information.

SMTP. Simple Mail Transfer Protocol.

SNA. Systems Network Architecture.

SNALINK. SNA Network Link.

SNA Network Link. An SNA network link function of TCP/IP for VM and MVS hosts running TCP/IP to communicate through an existing SNA backbone.

SNMP. Simple Network Management Protocol.

SOA. Start of authority record.

socket. An endpoint for communication between processes or applications. A pair consisting of TCP port and IP address, or UDP port and IP address.

socket address. An address that results when the port identification number is combined with an internet address.

socket interface. An application interface that allows users to write their own applications to supplement those supplied by TCP/IP.

SPOOL. Simultaneous peripheral operations online.

spooling. The processing of files created by or intended for virtual readers, punches, and printers. The spool files can be sent from one virtual device to another, from one virtual machine to another, and to read devices.

SQL. Structured Query Language.

SQL/DS. Structured Query Language/Data System.

SSL. Secure Sockets Layer. Provides the secure (encrypted) communication between a remote client and a TCP/IP server.

start of authority record (SOA). In the Domain Name System, the resource record that defines a zone.

stream. A continuous sequence of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

Structured Query Language (SQL). Fourth generation English-like programming language used to perform queries on relational databases.

Structured Query Language/Data System (SQL/DS). An IBM relational database management system for the VM and VSE operating systems.

Structure for Management Information (SMI). The rules used to define the objects that can be accessed through a network management protocol. See also *MIB*.

subagent. In the SNMP architecture, a subagent provides an extension to the utility provided by the SNMP agent.

subdirectory. A directory contained within another directory in a file system hierarchy.

subnet. A networking scheme that divides a single logical network into smaller physical networks to simplify routing.

subnet address. The portion of the host address that identifies a subnetwork.

subnet mask. A mask used in the IP protocol layer to separate the subnet address from the host portion of the address.

subnetwork. Synonymous with *subnet*.

subsystem. A secondary or subordinate system, usually capable of operating independent of, or asynchronously with, a controlling system.

SYNC. Synchronous.

synchronous (SYNC). Pertaining to two or more processes that depend on the occurrences of a specific event such as common timing signal. Occurring with a regular or predictable time relationship. See *asynchronous*.

synchronous data link control (SDLC). A data link over which communication is conducted using the synchronous data protocol.

Systems Application Architecture (SAA). A formal set of rules that enables applications to be run without modification in different computer environments.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

T

TALK. An interactive messaging system that sends messages between the local host and a foreign host.

TCP. Transmission Control Protocol.

TCP/IP. Transmission Control Protocol/Internet Protocol.

Telnet. The Terminal Emulation Protocol, a TCP/IP application protocol for remote connection service. Telnet allows a user at one site to gain access to a foreign host as if the user's terminal were connected directly to that foreign host.

terminal emulator. A program that imitates the function of a particular kind of terminal.

Terminate and Stay Resident (TSR) program. A TSR is a program that installs part of itself as an extension of DOS when it is executed.

TFTPD. Trivial File Transfer Protocol Daemon.

ticket. Encrypted information obtained from a Kerberos authentication server or a ticket-granting server. A ticket authenticates a user and, in conjunction with an authenticator, serves as permission to access a service when presented by the authenticated user.

ticket-granting server. Grants Kerberos tickets to authenticated users as permission to access an end-service.

Time Sharing Option (TSO). An operating system option; for System/370 system, the option provides interactive time sharing from remote terminals

time stamp. To apply the current system time. The value on an object that is an indication of the system time at some critical point in the history of the object. In query, the identification of the day and time when a query report was created that query automatically provides on each report.

TN3270. An informally defined protocol for transmitting 3270 data streams over Telnet.

token. In a local network, the symbol of authority passed among data stations to indicate the station temporarily in control of the transmission medium.

token-bus. See *bus topology*.

token ring. As defined in IEEE 802.5, a communication method that uses a token to control access to the LAN. The difference between a token bus and a token ring is that a token-ring LAN does not use a master controller to control the token. Instead, each computer knows the address of the computer that should receive the token next. When a computer with the token has nothing to transmit, it passes the token to the next computer in line.

token-ring network. A ring network that allows unidirectional data transmission between data stations by a token-passing procedure over one transmission medium, so that the transmitted data returns to the transmitting station.

Transmission Control Protocol (TCP). The TCP/IP layer that provides reliable, process-to-process data stream delivery between nodes in interconnected computer networks. TCP assumes that IP (Internet Protocol) is the underlying protocol.

Transmission Control Protocol/Internet Protocol (TCP/IP). A suite of protocols designed to allow communication between networks regardless of the technologies implemented in each network.

transport layer. Layer 4 of the Open Systems Interconnection (OSI) model; it defines protocols governing message structure and some error checking.

TRAP. An unsolicited message that is sent by an SNMP agent to an SNMP network management station.

Trivial File Transfer Protocol Daemon (TFTPD). The TFTP daemon (TFTPD server) transfers files between the Byte File System (BFS) and TFTP clients. TFTPD supports access to files maintained in a BFS directory structure that is mounted.

TSO. Time Sharing Option.

TSR. Terminate and stay resident. TSR usually refers to a terminate-and-stay-resident program.

U

UDP. User Datagram Protocol.

user. A function that uses the services provided by a server. A host can be a user and a server at the same time. See *client*.

User Datagram Protocol (UDP). A datagram level protocol built directly on the IP layer. UDP is used for application-to-application programs between TCP/IP hosts.

user exit. A point in an IBM-supplied program at which a user routine may be given control.

user profile. A description of a user, including user ID, user name, defaults, password, access authorization, and attributes.

V

virtual address. The address of a location in virtual storage. A virtual address must be translated into a real address to process the data in processor storage.

Virtual Machine (VM). Licensed software whose full name is Virtual Machine/Enterprise Systems Architecture (VM/ESA) It is a software operating system that manages the resources of a real processor to provide virtual machines to end users. It includes time-sharing system control program (CP), the conversational monitor system (CMS), the group control system (GCS), and the dump viewing facility (DVF).

Virtual Machine Communication Facility (VMCF). A connectionless mechanism for communication between address spaces.

Virtual Machine/System Product (VM/SP). An IBM-licensed program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a *real* machine.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, not by the actual number of main storage locations.

Virtual Telecommunications Access Method (VTAM). An IBM-licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VM. Virtual Machine.

VMCF. Virtual Machine Communication Facility.

VM/ESA. Virtual Machine/Enterprise System Architecture

VMSES/E. Virtual Machine Serviceability Enhancements Staged/Extended.

VTAM. Virtual Telecommunications Access Method.

W

WAN. Wide area network.

well-known port. A port number that has been preassigned for specific use by a specific protocol or application. Clients and servers using the same protocol communicate over the same well-known port.

wide area network (WAN). A network that provides communication services to a geographic area larger than that served by a local area network.

widget. The basic data type of the X Window System Toolkit. Every widget belongs to a widget class that contains the allowed operations for that corresponding class.

window. An area of the screen with visible boundaries through which a panel or portion of a panel is displayed.

working directory. The directory in which an application program is found. The working directory becomes the current directory when the application is started.

X

X Client. An application program which uses the X protocol to communicate windowing and graphics requests to an X Server.

XDR. eXternal Data Representation.

XEDIT. The CMS facility, containing the XEDIT command and XEDIT subcommands and macros, that lets a user create, change, and manipulate CMS files.

X Server. A program which interprets the X protocol and controls one or more screens, a pointing device, a keyboard, and various resources associated with the X Window System, such as Graphics Contexts, Pixmaps, and color tables.

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

X Window System API. An application program interface designed as a distributed, network-transparent, device-independent, windowing and graphics system.

X Window System Toolkit. Functions for developing application environments.

X.25. A CCITT communication protocol that defines the interface between data terminal equipment and packet switching networks.

X.25 NCP Packet Switching Interface (X.25 NPSI). An IBM-licensed program that allows users to communicate over packet switched data networks that have interfaces complying with Recommendation X.25 (Geneva** 1980) of the CCITT. It allows SNA programs to communicate with SNA equipment or with non-SNA equipment over such networks.

Z

ZAP. To modify or dump an individual text file/data set using the ZAP command or the ZAPTEXT EXEC.

ZAP disk. The virtual disk in the VM operating system that contains the user-written modifications to VTAM code.

zone. In the Domain Name System, a zone is a logical grouping of domain names that is assigned to a particular organization. Once an organization controls its own zone, it can change the data in the zone, add new tree sections connected to the zone, delete existing nodes, or delegate new subzones under its zone.

Bibliography

This bibliography lists the publications that provide information about your z/VM system. The z/VM library includes z/VM base publications, publications for additional facilities included with z/VM, and publications for z/VM optional features. For abstracts of z/VM publications and information about current editions and available publication formats, see *z/VM: General Information*.

z/VM Base Publications

Evaluation

- *z/VM: Licensed Program Specifications*, GC24-5943
- *z/VM: General Information*, GC24-5944

Installation and Service

- *z/VM: Installation Guide*, GC24-5945
- *z/VM: Service Guide*, GC24-5946
- *z/VM: VMSES/E Introduction and Reference*, GC24-5947

Planning and Administration

- *z/VM: Planning and Administration*, SC24-5948
- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-5949
- *z/VM: Migration Guide*, GC24-5928
- *VM/ESA: REXX/EXEC Migration Tool for VM/ESA*, GC24-5752
- *z/VM: Running Guest Operating Systems*, SC24-5950
- *VM/ESA: Connectivity Planning, Administration, and Operation*, SC24-5756
- *z/VM: Group Control System*, SC24-5951
- *z/VM: Performance*, SC24-5952

Customization

- *z/VM: CP Exit Customization*, SC24-5953

Operation

- *z/VM: System Operation*, SC24-5954
- *z/VM: Virtual Machine Operation*, SC24-5955

Application Programming

- *z/VM: CP Programming Services*, SC24-5956
- *z/VM: CMS Application Development Guide*, SC24-5957
- *z/VM: CMS Application Development Guide for Assembler*, SC24-5958
- *z/VM: CMS Callable Services Reference*, SC24-5959
- *z/VM: CMS Macros and Functions Reference*, SC24-5960
- *z/VM: CMS Application Multitasking*, SC24-5961
- *VM/ESA: REXX/VM Primer*, SC24-5598
- *z/VM: REXX/VM User's Guide*, SC24-5962
- *z/VM: REXX/VM Reference*, SC24-5963
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-5976
- *z/VM: OpenExtensions User's Guide*, SC24-5977
- *z/VM: OpenExtensions Command Reference*, SC24-5978
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-5979
- *z/VM: OpenExtensions Callable Services Reference*, SC24-5980
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-5964
- *z/VM: Enterprise Systems Architecture/Extended Configuration Principles of Operation*, SC24-5965
- *C for VM/ESA: Library Reference*, SC23-3908
- *OS/390: DFSMS Program Management*, SC27-0806
- *z/VM: Program Management Binder for CMS*, SC24-5934
- *Debug Tool User's Guide and Reference*, SC09-2137
- *External Security Interface (RACROUTE) Macro Reference for MVS and VM*, GC28-1366
- *VM/ESA: Programmer's Guide to the Server-Requester Programming Interface for VM*, SC24-5455
- *VM/ESA: CPI Communications User's Guide*, SC24-5595
- *Common Programming Interface Communications Reference*, SC26-4399
- *Common Programming Interface Resource Recovery Reference*, SC31-6821

End Use

- *z/VM: CP Command and Utility Reference*, SC24-5967
- *VM/ESA: CMS Primer*, SC24-5458
- *z/VM: CMS User's Guide*, SC24-5968
- *z/VM: CMS Command Reference*, SC24-5969
- *z/VM: CMS Pipelines User's Guide*, SC24-5970
- *z/VM: CMS Pipelines Reference*, SC24-5971
- *CMS/TSO Pipelines: Author's Edition*, SL26-0018
- *z/VM: XEDIT User's Guide*, SC24-5972
- *z/VM: XEDIT Command and Macro Reference*, SC24-5973
- *z/VM: Quick Reference*, SC24-5986

Diagnosis

- *z/VM: System Messages and Codes*, GC24-5974
- *z/VM: Diagnosis Guide*, GC24-5975
- *z/VM: VM Dump Tool*, GC24-5887
- *z/VM: Dump Viewing Facility*, GC24-5966

Publications for Additional Facilities

DFSMS/VM®

- *VM/ESA: DFSMS/VM Function Level 221 Planning Guide*, GC35-0121
- *VM/ESA: DFSMS/VM Function Level 221 Installation and Customization*, SC26-4704
- *VM/ESA: DFSMS/VM Function Level 221 Storage Administration Guide and Reference*, SH35-0111
- *VM/ESA: DFSMS/VM Function Level 221 Removable Media Services User's Guide and Reference*, SC35-0141
- *VM/ESA: DFSMS/VM Function Level 221 Messages and Codes*, SC26-4707
- *VM/ESA: DFSMS/VM Function Level 221 Diagnosis Guide*, LY27-9589

OSA/SF

- *S/390: Planning for the S/390 Open Systems Adapter (OSA-1, OSA-2) Feature*, GC23-3870
- *VM/ESA: Open Systems Adapter Support Facility User's Guide for OSA-2*, SC28-1992
- *S/390: Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7403

Language Environment®

- *Language Environment for OS/390 & VM: Concepts Guide*, GC28-1945
- *Language Environment for OS/390 & VM: Migration Guide*, SC28-1944
- *Language Environment for OS/390 & VM: Programming Guide*, SC28-1939
- *Language Environment for OS/390 & VM: Programming Reference*, SC28-1940
- *Language Environment for OS/390 & VM: Writing Interlanguage Communication Applications*, SC28-1943
- *Language Environment for OS/390 & VM: Debugging Guide and Run-Time Messages*, SC28-1942

Publications for Optional Features

CMS Utilities Feature

- *VM/ESA: CMS Utilities Feature*, SC24-5535

TCP/IP Feature for z/VM

- *z/VM: TCP/IP Level 3A0 Planning and Customization*, SC24-5981
- *z/VM: TCP/IP Level 3A0 User's Guide*, SC24-5982
- *z/VM: TCP/IP Level 3A0 Programmer's Reference*, SC24-5983
- *z/VM: TCP/IP Level 3A0 Messages and Codes*, GC24-5984
- *z/VM: TCP/IP Level 3A0 Diagnosis Guide*, GC24-5985

OpenEdition® DCE Feature for VM/ESA®

- *OpenEdition DCE for VM/ESA: Introducing the OpenEdition Distributed Computing Environment*, SC24-5735
- *OpenEdition DCE for VM/ESA: Planning*, SC24-5737
- *OpenEdition DCE for VM/ESA: Configuring and Getting Started*, SC24-5734
- *OpenEdition DCE for VM/ESA: Administration Guide*, SC24-5730
- *OpenEdition DCE for VM/ESA: Administration Reference*, SC24-5731
- *OpenEdition DCE for VM/ESA: Application Development Guide*, SC24-5732

- *OpenEdition DCE for VM/ESA: Application Development Reference*, SC24-5733
- *OpenEdition DCE for VM/ESA: User's Guide*, SC24-5738
- *OpenEdition DCE for VM/ESA: Messages and Codes*, SC24-5736

LANRES/VM

- *LAN Resource Extension and Services/VM: Licensed Program Specifications*, GC24-5617
- *LAN Resource Extension and Services/VM: General Information*, GC24-5618
- *LAN Resource Extension and Services/VM: Guide and Reference*, SC24-5622

CD-ROM

The following CD-ROM contains all the IBM libraries that are available in IBM BookManager® format for current VM system products and current IBM licensed programs that run on VM. It also contains PDF versions of z/VM publications and publications for some related IBM licensed programs.

- *Online Library Omnibus Edition: VM Collection*, SK2T-2067

Note: Only unlicensed publications are included.

Other TCP/IP Related Publications

This section lists other publications, outside the z/VM 3.1.0 library, that you may find helpful.

- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *TCP/IP Illustrated, Volume 1: The Protocols*, SR28-5586
- *Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture*, SC31-6144
- *Internetworking With TCP/IP Volume II: Implementation and Internals*, SC31-6145
- *Internetworking With TCP/IP Volume III: Client-Server Programming and Applications*, SC31-6146
- *DNS and BIND in a Nutshell*, SR28-4970
- "MIB II Extends SNMP Interoperability," C. Vanderberg, *Data Communications*, October 1990.
- *"Network Management and the Design of SNMP,"* J.D. Case, J.R. Davin, M.S. Fedor, M.L. Schoffstall.
- *"Network Management of TCP/IP Networks: Present and Future,"* A. Ben-Artzi, A. Chandna, V. Warrier.
- "Special Issue: Network Management and Network Security,"*ConneXions-The Interoperability Report*, Volume 4, No. 8, August 1990.
- *The Art of Distributed Application: Programming Techniques for Remote Procedure Calls*, John R. Corbin, Springer-Verlog, 1991.
- *The Simple Book: An Introduction to Management of TCP/IP-based Internets*, Marshall T Rose, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Index

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

z/VM
TCP/IP Level 3A0
Planning and Customization
Version 3 Release 1.0

Publication No. SC24-5981-00

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
SC24-5981-00

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 G60G
1701 North Street
Endicott, New York
13760-5553

Fold and Tape **Please do not staple** Fold and Tape

SC24-5981-00

Cut or Fold
Along Line

File Number: S370/4300/30XX-50
Program Number: 5654-A17

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC24-5981-00

Spine information:

z/VM TCP/IP Level 3A0 Planning and Customization Version 3 Release 1.0