z/OS Communications Server

IP Configuration Guide

Version 1 Release 1

SC31-8775-00

z/OS Communications Server

IP Configuration Guide

Version 1 Release 1

SC31-8775-00

Note:
Before using this information and the product it supports, be sure to read the general information under “Appendix E. Notices” on page 585.

First Edition (March 2001)

This edition applies to Version 1 Release 1 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

IBM Corporation
Software Reengineering
Department G7IA/ Bldg 503
Research Triangle Park, NC 27709-9990
U.S.A.

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

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

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

World Wide Web:
http://www.ibm.com/servers/eserver/zseries/zos/

IBMLink:
CIBMORCF at RALVM17

IBM Mail Exchange:
tkinlaw@us.ibm.com

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

Contents

Figures

Tables

About This Book

This book contains guidance material to enable you to configure IP address spaces, servers, and applications for z/OS Communications Server (z/OS CS). This volume is part of a two-volume set:

- *z/OS Communications Server: IP Configuration Guide*, which contains concepts and guidance, explaining an overall approach to IP configuration.
- *z/OS Communications Server: IP Configuration Reference*, which describes parameters and options, syntax of statements and common IP commands used during configuration.

For detailed information about configuration-related data sets and statements, refer to *z/OS Communications Server: IP Configuration Reference*.

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

z/OS CS is an integral part of the z/OS V1R1 family of products. For an overview and mapping of the documentation available for z/OS V1R1, refer to the *z/OS Information Roadmap*.

Who Should Use This Book

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

How to Use This Book

Use this book to perform the following tasks:

- Configure z/OS CS
- Customize and administer z/OS CS

How This Book Is Organized

This book is divided into three parts:

Part 1. “Configuring the Base TCP/IP System”, contains general information about configuring the base TCP/IP stack and routing.

Part 2. “Configuring the Servers”, contains chapters that explain the server configuration process for z/OS CS.

Part 3. “Appendixes”, provides additional information.

Where to Find More Information

The bibliography at the end of this book describes the books in the z/OS CS library, arranged according to task. The bibliography also lists the files and order numbers of books related to this book, or cited by name in this book.

Most licensed books were declassified in OS/390® V2R4 and are now included in the z/OS Online Library Collection, SK2T-6700. The remaining licensed books appear in unencrypted BookManager® softcopy and PDF form on the z/OS Licensed Product Library, LK2T-2499.

Where to Find Related Information on the Internet

You might find the following information helpful.

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

Home Page	Web address
z/OS	http://www.ibm.com/servers/eserver/zseries/zos/
z/OS Internet Library	http://www.ibm.com/servers/eserver/zseries/zos/bkserv/
IBM Communications Server product	http://www.software.ibm.com/network/commserver/
IBM Communications Server support	http://www.software.ibm.com/network/commserver/support/
IBM Systems Center publications	http://www.redbooks.ibm.com/
IBM Systems Center flashes	http://www-1.ibm.com/support/techdocs/atsmastr.nsf
VTAM and TCP/IP	http://www.software.ibm.com/network/commserver/about/csos390.html
IBM	http://www.ibm.com

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

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

You can look up any message by number with the LookAt online message help facility. You can access LookAt and instructions for it's use at the following Web address:

http://www.s390.ibm.com/os390/bkserv/lookat/lookat.html

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

How to Contact IBM® Service

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

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

Part 1. Base TCP/IP System

Chapter 1. Configuration Overview

The objective of this chapter is to help you be better prepared for installation related activities. It is important to understand the terms, relationships, and dependencies presented in this chapter as a prerequisite to installation and customization.

After reading this chapter, you will be familiar with:

- z/OS UNIX System Services concepts
- Differences between HFS files and MVS data sets
- TCP/IP applications rewritten for z/OS UNIX System Services (z/OS UNIX)
- Configuration files and their search orders
- MVS Considerations
- Accounting and security issues for the more commonly used daemons
- Considerations for multiple instances of TCP/IP
- Enterprise Extender considerations
- Virtual IP Address (VIPA) considerations

z/OS UNIX System Services (z/OS UNIX) Concepts

Beginning with MVS/ESA™ Version 4.3 a new type of application program interface was added to the MVS platform with the intent of integrating a UNIX operating system into MVS. Both a C programming API and an interactive environment called the shell were defined to interoperate with UNIX-style files, called Hierarchical File Systems (HFS). Over time, other organizations developed approaches to working with UNIX on various platforms until finally an organization named X/Open documented standards of what to implement for UNIX interfaces in a series of guides published as the X/Open Portability Guides (XPG). X/Open now owns the term UNIX and certifies different implementations of UNIX according to the UNIX definitions contained in XPG 4.2. In 1996, OS/390 OpenEdition® was awarded UNIX 95 brand certification, thus confirming that it is compliant with all current open industry standards.

Note: In 1998, IBM changed the name OS/390 OpenEdition to OS/390 UNIX System Services.

z/OS UNIX System Services or z/OS UNIX is the z/OS or MVS implementation of UNIX as defined by X/Open in the XPG 4.2. z/OS UNIX coexists with traditional MVS functions and traditional MVS file types (partitioned data sets, sequential files, and so on). It concurrently allows access to HFS files and to UNIX utilities and commands by means of application programming interfaces (APIs) and the interactive SHELL environment. MVS offers two variants of the UNIX SHELL environment:

- The OMVS shell, much like a native UNIX environment
- The ISHELL, an ISPF interface with access to menu-driven command interfaces

With the APIs, programs can run in any environment including batch jobs, in jobs submitted by TSO/E interactive users, and in most other started tasks, or in any other MVS application task environment. The programs can request:

- Only MVS services
- Only z/OS UNIX services
- Both MVS and z/OS UNIX services

The shell interface is an execution environment analogous to TSO/E, with a programming language of shell commands analogous to Restructured eXtended eXecutor (REXX) language. The shell support consists of:

- Programs that are run interactively by shell users
- Shell commands and scripts that are run interactively by shell users
- Shell commands and scripts that are run as batch jobs

Prior to OS/390 V2R5, OS/390 UNIX required APPC/MVS for programs issuing the fork() or spawn() function of OpenEdition callable services. APPC/MVS is no longer required for this purpose. Forked and spawned address spaces are now implemented in z/OS for UNIX processing by the Work Load Manager (WLM) component of MVS.

For a fork(), the system copies one process, called the parent process, into a new process, called the child process, and places the child process in a new address space, the forked address space.

Spawn() also starts a new process in a new address space. Unlike a fork(), in a spawn() call the parent process specifies a name of a program to start the child process.

The types of processes can be:

- User processes, which are associated with a user
- Daemon processes, which perform continuous or periodic functions, such as a Web server

  Daemons are programs that are typically started when the operating system is initialized and remain active to perform standard services. Some programs that initialize processes for users are considered daemons, even though these daemons are not long-running processes. Examples of daemons provided by z/OS UNIX are *cron*, which starts applications at specific times, and *inetd*, which starts applications on demand.

A user or daemon process can have one or more threads. A thread is a single flow of control within a process. Application programmers create multiple threads to structure an application in independent sections that can run in parallel for more efficient use of system resources.

Overview of Data Sets and HFS Files

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

Some data sets and HFS files have special importance because of their function. For example, certain data sets and HFS files are used when configuring the TCP/IP environment. Other data sets are used by the Telnet server (Telnet daemon) when performing specific communication functions. See Table 1 on page 8 for descriptions of the data sets and HFS files necessary for configuring the TCP/IP environment and the search orders used to find them. A search order can include both HFS files and data sets, and these data sets and HFS files will be collectively referred to as the *configuration files* in this section.

Note: Not all applications support HFS files.

Hierarchical File System Concepts

The Hierarchical File System lets you to set up a file hierarchy that consists of:

- HFS **files**, which contain data or programs. A file containing a load module, shell script, or REXX program is called an *executable file*. Files are kept in directories.
- **Directories** that contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside-down tree, with root directory at the top and the branches at the bottom. The **root** is the first directory for the file system at the peak of the tree and is designated by a slash (/).
- Additional local or remote **file systems**, that are mounted on directories of the root file system or of additional file systems.
- Lastly, the HFS also includes named pipes, links, and other UNIX items. One of these is character special files like /dev/console that are used by applications like syslogd. Refer to *z/OS UNIX System Services Planning* for more information about UNIX items like character special files.

To the z/OS system, the file hierarchy is a collection of hierarchical file system (HFS) data sets. Each HFS data set is a mountable file system. The root file system is the first file system mounted. Subsequent file systems can be logically mounted on a directory within the root file system or on a directory within any mounted file system.

Except for the direction of the slashes, the hierarchical file system is similar to a Disk Operating System (DOS) or an OS/2® file system.

Each *mountable* file system resides in a hierarchical file system (HFS) data set on direct access storage. DFSMS/MVS® manages the HFS data sets and the physical files.

The Root File System

The root system is the starting point for the overall HFS file structure. It contains the root directory and any related HFS files or subdirectories. The root file system is created as part of the installation process, either the SERVERPAC method or CPBDO, when you install z/OS.

TCP/IP Applications Rewritten for z/OS UNIX

If you are not currently converting to z/OS UNIX applications and you prefer to keep a TCP/IP MVS appearance in your day-to-day operations, some basic changes have been made which might result in changes to your configuration. Some of the TCP/IP MVS applications shipped in previous releases were rewritten as OS/390 UNIX applications in OS/390 V2R5. These include, but are not limited to, the following:

- FTP client and server
- OROUTED
- DNS
- SNMP

For more information on each of these topics, see the appropriate server chapters in this book. In addition, since OS/390 V2R5, several new servers, such as the policy agent, have been written as z/OS UNIX applications.

Information on these applications can also be found in *z/OS Communications Server: IP Migration*, *z/OS Communications Server: IP User's Guide*, and *z/OS Communications Server: IP Configuration Reference*.

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

- z/OS UNIX applications use the LE resolver (see Figure 1 on page 16), which uses the z/OS UNIX search order (see Table 2 on page 17) to find the data set prefix (or HLQ) for accessing files, looking up host names and addresses, local port numbers, and protocol numbers.
- TCP/IP z/OS UNIX applications can include HFS files in the search order for specific client and server configuration files.

Understanding Search Orders of Configuration Information

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

It is important to note that the z/OS CS environment consists of the TCP/IP address space, z/OS CS applications, and the TCP/IP MVS applications. The TCP/IP address space functions are also referred to as the *stack*. The TCP/IP z/OS UNIX applications refer to those applications using the z/OS UNIX socket API. The TCP/IP MVS applications refer to those applications written to the MVS APIs (for example, C, Sockets-Extended, CICS®, IMS™, and REXX). The TCP/IP stack and both sets of applications have some common (or global) configuration files, but they also use configuration files that are different.

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

Configuration Data Set Naming Conventions

When searching for configuration files, you can explicitly tell TCP/IP where most configuration files are by using DD statements in the JCL procedures or by setting environment variables. Otherwise, you can let TCP/IP dynamically determine the location of the configuration files, based on search orders shown in Table 1 on page 8.

For example, in Table 1 on page 8, for the FTP server application, if the installation did not code the //SYSFTPD DD statement, the FTP server would search for *jobname*.FTP.DATA, then file /etc/ftp.data, then data set SYS1.TCPPARMS(FTPDATA), and finally *hlq*.FTP.DATA.

Dynamic Data Set Allocation

TCP/IP makes extensive use of dynamically allocated data sets using the MVS dynamic data set allocation function to search for configuration files. Multiple versions of a configuration data set can exist, each having a different high-level qualifier or middle-level qualifier. The search order for any configuration file will determine which data set is found and used.

High-Level Qualifier: TCP/IP is distributed with a default high-level qualifier (HLQ) of *TCPIP*. To override the default HLQ used by dynamic data set allocation, specify the DATASETPREFIX statement in the TCPIP.DATA configuration file. For most configuration files, the DATASETPREFIX value is used as the high-level qualifier of the data set name in the last step in the search order. Note that the DATASETPREFIX value is not used as the high-level qualifier of the data set name used as the last step in the search order for the PROFILE.TCPIP and TCPIP.DATA configuration files.

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

- Node name

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

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

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

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

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

- User ID or job name
- DATASETPREFIX value

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

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

- *userid.yyyy.zzzz*

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

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

- *jobname.yyyy.zzzz*

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

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

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

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

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

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

Table 1 lists the configuration data sets used by the TCP/IP servers and functions. It includes the name of the sample and the usage of the data set.

Table 1. TCP/IP Configuration Data Sets

Data Set/Search Order	Copied From	Usage
ADM@ACL.ADD	SEZAINST(ADMADD)	Used by the Kerberos server, authorizes remote administrators to add database entries.
ADM@ACL.GET	SEZAINST(ADMGET)	Used by the Kerberos server, authorizes remote administrators to query database entries.
ADM@ACL.MOD	SEZAINST(ADMMOD)	Used by the Kerberos server, authorizes remote administrators to modify database entries.
ETC.PROTO	usr/lpp/tcpip/samples/protocol	Used to map types of protocol to integer values to determine the availability of the specified protocol. Required by several z/OS CS components. **Note:** The search order depends on the type of application (z/OS UNIX or native MVS).
ETC.RPC	SEZAINST(ETCRPC)	Defines RPC applications to the Portmapper function.

Table 1. TCP/IP Configuration Data Sets (continued)

Data Set/Search Order	Copied From	Usage
ETC.SERVICES	usr/lpp/tcpip/samples/services	Establishes port numbers for servers using TCP and UDP. Required for z/OS UNIX SNMP, OROUTED, and OMPROUTE (if the RIP protocol is used). **Note:** The search order depends on the type of application (z/OS UNIX or native MVS).
FTP.DATA 1. //SYSFTPD 2. *userid/jobname*.FTP.DATA 3. /etc/ftp.data 4. SYS1.TCPPARMS(FTPDATA) 5. *hlq*.FTP.DATA	SEZAINST(FTCDATA) for the client and (FTPSDATA) for the server	Overrides default FTP client and server parameters for the FTP server.
HOSTS.LOCAL (or /etc/hosts)	SEZAINST(HOSTS)	Input data set to MAKESITE for generation of HOSTS.ADDRINFO and HOSTS.SITEINFO.
KRBCONF	SEZAINST(KRBCONF)	Identifies the hosts running the Kerberos authentication server.
LPD.CONFIG	SEZAINST(LPDDATA)	Configures the Line Printer Daemon for the Remote Print Server.
LU62CFG	SEZAINST(LU62CFG)	Provides configuration parameters for the SNALINK LU6.2 interface.
MASTER.DATA	No sample provided	DNS database input required for authoritative name servers.
MIBS.DATA • The name of an HFS file or an MVS file specified by the MIBS_DATA environment variable • /etc/mibs.data HFS file	No sample provided.	Defines ″textual″ names for MIB objects for the osnmp command.
NPSIDATE	SEZAINST(NPSIDATE)	Operates the TCP/IP X.25 NCP Packet Switching Interface.
NPSIFC	SEZAINST(NPSIFC)	Supports GATE MCH Fast Connect option for X.25 NCP Packet Switching Interface.
NPSIGATE	SEZAINST(NPSIGATE)	Supports GATE MCHs for X.25 NCP Packet Switching Interface.
OSNMPD.CONF 1. /etc/osnmp.conf 2. /etc/snmpv2.conf	/usr/lpp/tcpip/samples/snmpv2.conf	Defines target host security parameters for the osnmp command.

Table 1. TCP/IP Configuration Data Sets (continued)

Data Set/Search Order	Copied From	Usage
OSNMPD.DATA 1. The name of an HFS file or MVS file specified by the OSNMPD_DATA environment variable. 2. /etc/osnmpd.data HFS file. 3. The data set specified on the OSNMPD DD statement in the agent procedure. 4. *jobname*.OSNMPD.DATA, where *jobname* is the name of the job used to start the SNMP agent. 5. SYS1.TCPPARMS(OSNMPD). 6. *hlq*.OSNMPD.DATA, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used. **Note:** The first file found in the search order is used.	/usr/lpp/tcpip/samples/osnmpd.data	Used by SNMP for setting values for selected MIB objects.
PAGENT.CONF 1. Specified file or data set 2. /etc/pagent.conf 3. hlq.PAGENT.CONF	SEZAINST(PAGENT)	Defines Policy Agent configuration parameters and optionally defines service policies (rules and actions).
PROFILE.TCPIP 1. //PROFILE 2. *job_name.node_name*.TCPIP 3. *hlq.node_name*.TCPIP 4. *job_name*.PROFILE.TCPIP 5. *hlq*.PROFILE.TCPIP	SEZAINST(SAMPPROF)	Provides TCP/IP initialization parameters and specifications for network interfaces and routing.
PW.SRC 1. The name of an HFS file or an MVS file specified by the PW_SRC environment variable. 2. /etc/pw.src HFS file. 3. The data set specified on SYSPWSRC DD statement in the agent procedure. 4. *jobname*.PW.SRC, where *jobname* is the name of the job used to start the SNMP agent. 5. SYS1.TCPPARMS(PWSRC). 6. *hlq*.PW.SRC, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used. **Note:** The first file found in the search order is used.	No sample provided	Defines a list of community names used when accessing objects on a destination SNMP agent.

Table 1. TCP/IP Configuration Data Sets (continued)

Data Set/Search Order	Copied From	Usage
RSVPD.CONF 1. Specified file or data set 2. /etc/rsvpd.conf 3. hlq.RSVPD.CONF	SEZAINST(RSVPDCF)	Defines RSVP Agent configuration parameters.
SNMPD.BOOTS 1. The name of an HFS file or an MVS file specified by the SNMPD_BOOTS environment variable. 2. /etc/snmpd.boots. **Note:** The first file found in the search order is used.	No sample provided	Defines the SNMP agent security and notification destinations. **Note:** If the SNMPD.BOOTS file is not provided, the SNMP agent creates the file. If multiple SNMPv3 agents are running on the same MVS image, use the environment variable to specify different SNMPD.BOOTS files for the different agents. For security reasons, ensure unique engine IDs are used for different SNMP agents.
SNMPD.CONF 1. The name of an HFS file or an MVS file specified by the SNMPD_CONF environment variable. 2. /etc/snmpd.conf. **Note:** The first file found in the search order is used.	/usr/lpp/tcpip/samples/snmpd.conf	Defines the SNMP agent security and notification destinations. **Note:** If the SNMPD.CONF file is found, the PW.SRC file and the SNMPTRAP.DEST files are not used.
SNMPTRAP.DEST 1. The name of an HFS file or an MVS file specified by the SNMPTRAP_DEST environment variable. 2. /etc/snmptrap.dest HFS file. 3. The data set specified on SNMPTRAP DD statement in the agent procedure. 4. *jobname*.SNMPTRAP.DEST, where *jobname* is the name of the job used to start the SNMP agent. 5. SYS1.TCPPARMS(SNMPTRAP). 6. *hlq*.SNMPTRAP.DEST, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used. **Note:** The first file found in the search order is used.	No sample provided.	Defines a list of managers to which the SNMP agent will send traps.
SMTPCONF	SEZAINST(SMTPCONF)	Provides configuration parameters for the Simple Mail Transfer Protocol
SMTPNOTE	SEZAINST(SMTPNOTE)	Defines note parameters for Simple Mail Transfer Protocol

Table 1. TCP/IP Configuration Data Sets (continued)

Data Set/Search Order	Copied From	Usage
TCPIP.DATA	SEZAINST(TCPDATA)	Provides parameters for TCP/IP client programs. **Note:** The search order depends on the type of application (z/OS UNIX or native MVS).
TNDBCSCN	SEZAINST(TNDBCSCN)	Provides configuration parameters for Telnet 3270 Transform support.
TRAPFWD.CONF 1. An HFS file or an MVS data set specified by the TRAPFWD_CONF environment variable. 2. /etc/trapfwd.conf. **Note:** The first file found in the search order is used.	No sample provided	Defines addresses to which the Trap Forwarder Daemon will forward traps. **Note:** If the environment variable is set and if the file specified by the environment variable is not found, the Trap Forwarder daemon will terminate.
VTAMLST	SEZAINST(VTAMLST)	Defines VTAM® applications and their characteristics. Entries required for Telnet, SNALINK LU0, SNALINK LU6.2, and X.25 NPSI Server.
X25CONF	SEZAINST(X25CONF)	Provides configuration parameters for the X.25 NCP Packet Switching Interface.
X25VSVC	SEZAINST(X25VSVC)	Provides switched virtual circuit configuration for the X.25 NCP Packet Switching Interface.

Configuration Files for the TCP/IP Stack

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

PROFILE.TCPIP Search Order

During initialization of the TCP/IP stack, system operation and configuration parameters for the TCP/IP stack are read from the configuration file PROFILE.TCPIP. As shown in Table 1 on page 8, the search order used by the TCP/IP stack to find PROFILE.TCPIP involves both explicit and dynamic data set allocation as follows:

- //PROFILE DD DSN=*aaa.bbb.ccc(anyname)*
- *jobname.nodename*.TCPIP
- *hlq.nodename*.TCPIP
- *jobname*.PROFILE.TCPIP
- TCPIP.PROFILE.TCPIP

Note: Explicitly specifying the PROFILE DD statement in the TCPIPROC JCL is the recommended way to specify PROFILE.TCPIP. If this DD statement is present, the data set it defines is explicitly allocated by MVS and no dynamic

allocation is done. If this statement is not present, the search order continues to use dynamic allocation for the PROFILE.TCPIP.

Examples

The following examples show the search order used by TCP/IP to find the configuration file PROFILE.TCPIP. These examples use the sample TCP/IP started procedure, TCPIPROC, installed in the *hlq*.SEZAINST data set.

Example When DD Cards Are In Your TCP/IP Start-up Procedure: In this example, the PROFILE DD cards are specified as follows:

```
//TCPIP  PROC PARMS='CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//* SMP/E Distribution Name: EZAEB01G
//*
//*     5694-A01 (C) Copr. IBM Corp. 1991,2000.
//*     All rights reserved.
//*     US Government Users Restricted Rights -
//*     Use, duplication or disclosure restricted
//*     by GSA ADP Schedule Contract with IBM Corp.
//*     See IBM Copyright Instructions
//*
//TCPIP    EXEC PGM=EZBTCPIP,
//             PARM='&PARMS',
//             REGION=0K,TIME=1440
//*
:
//PROFILE DD DISP=SHR,DSN=MVSA.PROD.PARMS(PROFILE)
:
```

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

Example When No DD Cards Are In Your TCP/IP Start-up Procedure: In this example, the PROFILE DD statement is not specified:

```
//TCPIP  PROC PARMS='CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//* SMP/E Distribution Name: EZAEB01G
//*
//*     5694-A01 (C) Copr. IBM Corp. 1991,2000.
//*     All rights reserved.
//*     US Government Users Restricted Rights -
//*     Use, duplication or disclosure restricted
//*     by GSA ADP Schedule Contract with IBM Corp.
//*     See IBM Copyright Instructions
//*
//TCPIP    EXEC PGM=EZBTCPIP,
//             PARM='&PARMS',
//             REGION=0K,TIME=1440
//*
:
```

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

1. PROFILE DD

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

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

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

4. *jobname*.PROFILE.TCPIP

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

5. TCPIP.PROFILE.TCPIP

   TCPIP.PROFILE.TCPIP is searched last if necessary.

TCPIP.DATA Search Order

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

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

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

```
//TCPIP    PROC PARMS='CTRACE(CTIEZB00)'
//*
//* Communication Server/390
//*  SMP/E Distribution Name: EZAEB01G
//*
//*  Licensed Materials - Property of IBM
//*  "Restricted Materials of IBM"
//*  5694-A01
//*  (C) Copyright IBM Corp. 1991, 2000
//*  Status = CSV2R10
//*
//TCPIP    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//         PARM=('&PARMS',
//        'ENVAR("RESOLVER_CONFIG=//''TCPIVP.TCPPARMS(TCPDATA)''")')
//*
```

2. /etc/resolv.conf

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

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

   The //SYSTCPD DD card can be specified in TCPIPROC JCL.

4. *userid*.TCPIP.DATA, where *userid* is the user ID that is associated with the current security environment for the TCP/IP address space.

5. SYS1.TCPPARMS(TCPDATA)

6. *hlq*.TCPIP.DATA

   Remember that the default *hlq* distributed with TCP/IP is the string *TCPIP*. So, effectively, this is TCPIP.TCPIP.DATA in the search order. This has implications for installations using multiple TCP/IP address spaces.

If the SYSTCPD DD is used, it is only searched for if the RESOLVER_CONFIG environment variable is not set and the HFS file /etc/resolv.conf does not exist. For more information about a multiple TCP/IP instance environment, see "Considerations for Multiple Instances of TCP/IP" on page 37.

Note: The Telnet server runs in the TCP/IP address space. The Telnet host name mapping function does not use the RESOLVER_CONFIG TCPIP.DATA file, nor will it use the default HFS /etc/resolv.conf file. If Telnet host name mapping is used, the SYSTCPD DD or other default must be used to specify the resolver search order. If RESOLVER_CONFIG specifies an MVS data set, the SYSTCPD DD must specify the same MVS data set to ensure that the TCP/IP stack and the Telnet server use the same information.

Configuration Files for TCP/IP Applications

This section describes the configuration files that can be common to all TCP/IP applications (depending on the resolver that they use) and the search orders for those configuration files. Each application can also have its own configuration files that are specific to that application. For more information about specific configuration files, see the descriptions of the individual applications in “Part 2. Server Applications” on page 187.

Resolver Configuration Files

The resolver configuration files are used to resolve host names into IP addresses. As shown in Figure 1 on page 16, files called configuration files or data sets are used by the resolver component, which is part of the socket library. Although some shipped TCP/IP applications like onslookup have their own resolvers built in, TCP/IP MVS TCP/IP MVS sockets deliver the native MVS sockets resolver that is used with TCP/IP MVS APIs (C, Sockets Extended, CICS, IMS, REXX). Language Environment® for OS/390 V2R5 and later releases delivers another resolver that is used by all z/OS UNIX socket programs, referred to as the *z/OS UNIX Systems Services (z/OS UNIX) resolver*.

For example, if you run CICS or IMS applications, you would use the native MVS sockets resolver. UNIX applications such as *oping* or *onetstat* require the use of the z/OS UNIX resolver. Understanding the search orders used by these resolvers is key to setting up your system properly.

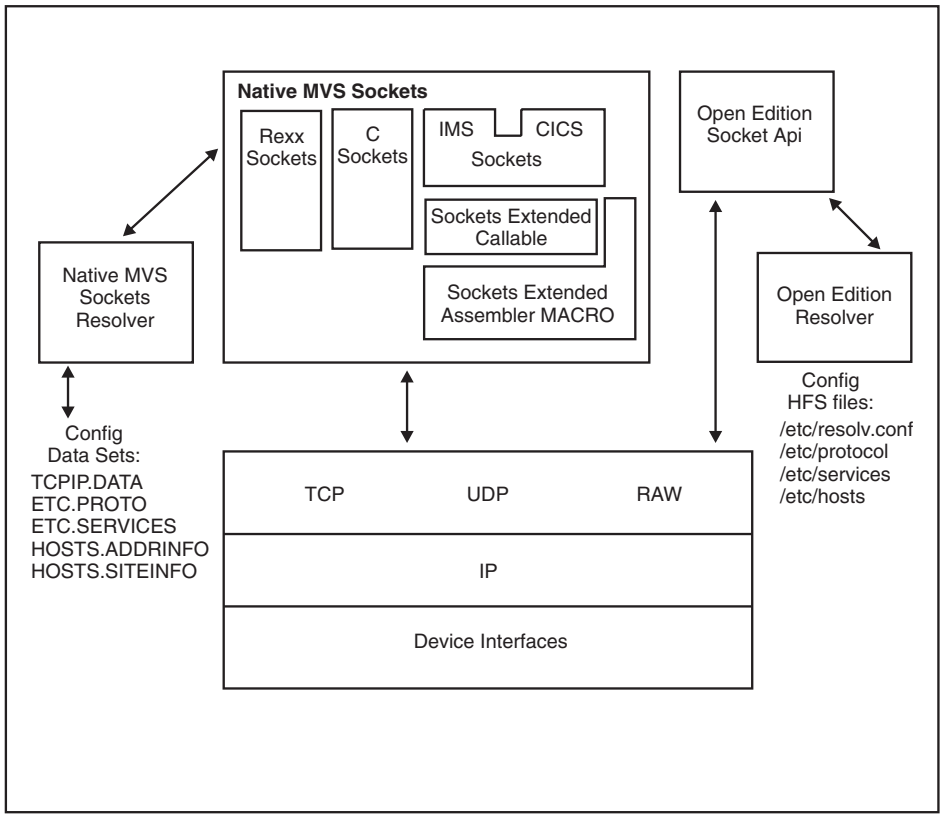


Figure 1. Resolver Components and Related Configuration Files in z/OS UNIX and TCP/IP Environments

There is a search order for each configuration (TCPIP.DATA vs. /etc/resolv.conf) data set; there is also a different hierarchy of possible locations for each data set or file.

For the native MVS sockets resolver, the TCPIP.DATA data set has the following search order:

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

For the z/OS UNIX Systems Services (z/OSUNIX) resolver, the base resolver configuration data set or file (the one that corresponds to TCPIP.DATA) has the following search order:

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

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

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

2. /etc/resolv.conf that resides in the HFS
3. The data set specified on the //SYSTCPD DD card

   DD names are not propagated from the parent process over the fork() or exec function calls. The allocation to SYSTCPD will not be available to the child process that is forked because DD allocations for the parent process are not inherited by the child. The only exception to this rule is a STEPLIB allocation.
4. userid.TCPIP.DATA, where *userid* is the user ID that is associated with the current security environment (address space or task/thread)
5. SYS1.TCPPARMS(TCPDATA)
6. TCPIP.TCPIP.DATA

Similar hierarchies exist for all the resolver configuration data sets or files as shown in Table 2.

Table 2. Resolver Configuration Data Sets and Files

Native MVS Sockets Resolver Search Order	**z/OS UNIX Resolver Search Order**
Base resolver configuration:	**Base resolver configuration**:
1. SYSTCPD DD-name 2. jobname.TCPIP.DATA or userid.TCPIP.DATA 3. SYS1.TCPPARMS(TCPDATA) 4. TCPIP.TCPIP.DATA	1. RESOLVER_CONFIG environment variable 2. /etc/resolv.conf 3. SYSTCPD DD-name 4. userid.TCPIP.DATA 5. SYS1.TCPPARMS(TCPDATA) 6. TCPIP.TCPIP.DATA
Local hosts tables:	**Local hosts tables**:
1. jobname.HOSTS.xxxxINFO or userid.HOSTS.xxxxINFO 2. *hlq*.HOSTS.xxxxINFO	1. X_SITE and X_ADDR environment variable 2. /etc/hosts 3. userid.HOSTS.xxxxINFO 4. *hlq*.HOSTS.xxxxINFO
Protocol information:	**Protocol information**:
1. jobname.ETC.PROTO or userid.ETC.PROTO 2. *hlq*.ETC.PROTO	1. /etc/protocol 2. userid.ETC.PROTO 3. *hlq*.ETC.PROTO
Service information:	**Service information**:
1. SERVICES DD-name 2. jobname.ETC.SERVICES or userid.ETC.SERVICES 3. *hlq*.ETC.SERVICES	1. /etc/services 2. userid.ETC.SERVICES 3. *hlq*.ETC.SERVICES
Translate table:	**Translate table**:
1. jobname.STANDARD.TCPXLBIN or userid.STANDARD.TCPXLBIN 2. *hlq*.STANDARD.TCPXLBIN	1. X_XLATE environment variable 2. userid.STANDARD.TCPXLBIN or jobname.STANDARD.TCPXLBIN 3. *hlq*.STANDARD.TCPXLBIN 4. Internal hard-coded translate table

Table 2. Resolver Configuration Data Sets and Files (continued)

Native MVS Sockets Resolver Search Order	z/OS UNIX Resolver Search Order
Note: *hlq.* comes from the DATASETPREFIX parameter in the base resolver configuration data set (TCPIP.DATA)	**Note:** *hlq.* comes from the DATASETPREFIX parameter in the base resolver configuration data set or file (TCPIP.DATA or /etc/resolv.conf) Where environment variables are used, they can either refer to an MVS data set name or to an HFS file name.
The FTP client, regardless of whether it is started from MVS/TSO or from z/OS UNIX, uses the z/OS UNIX resolver search order.	

Setting Environment Variables for Resolver Configuration Files

An environmental variable is an identifier used by the z/OS UNIX resolver like a variable in a program. In Table 2 on page 17 the following environmental variables appear:

RESOLVER_CONFIG
The resolver configuration data sets or files.

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

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

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

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

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

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

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

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

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

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

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

The following example shows an alternate method of accessing environment variables:

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

In this case, the environment variables will be read from the data set specified on the STDENV DD statement.

Understanding TCP/IP Data Set Names with z/OS UNIX Services

The z/OS UNIX socket runtime library (RTL) functions can use the following TCP/IP data sets:

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

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

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

TCPIP.DATA

This file is referenced to determine, among other things, the data set prefix (DATASETPREFIX keyword) to be used when trying to access the rest of the configuration files specified in this section. For the search order used to find the TCPIP.DATA configuration file, see Table 2 on page 17.

STANDARD.TCPXLBIN

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

The search order used to access this configuration file is:

1. The value of the environment variable X_XLATE

   The value of the environment variable is used to fopen() the configuration file. All z/OS UNIX I/O rules apply. For more information, refer to *z/OS C/C++ Programming Guide*.

2. *hlq*.STANDARD.TCPXLBIN

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

HOSTS.SITEINFO

This file supplies the information for the following four network host database functions:

- gethostbyname()
- sethostent()
- gethostent()
- endhostent()

Additionally, it supplies information for the network database function getnetbyname().

The search order used to access this configuration file is:

1. The value of the environment variable X_SITE

   The value of the environment variable is used to fopen() the configuration file. All z/OS UNIX I/O rules apply. For more information, refer to *z/OS C/C++ Programming Guide*.

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

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

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

HOSTS.ADDRINFO

This file supplies the information for the following four network database functions:

- getnetbyaddr()
- setnetent()
- getnetent()
- endnetent()

Additionally, it supplies information for the network host database function gethostbyaddr().

The search order used to access this configuration file is:

1. The value of the environment variable X_ADDR.

   The value of the environment variable is used to fopen() the configuration file. All z/OS UNIX I/O rules apply. For more information, refer to *z/OS C/C++ Programming Guide*.

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

4. *hlq*.HOSTS.ADDRINFO

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

ETC.PROTO

This file supplies the information for the following five protocol database functions:

- getprotobynumber()
- getprotobyname()
- setprotoent()
- getprotoent()
- endprotoent()

See *z/OS Communications Server: IP Configuration Reference* for more information.

The search order used to access this configuration file is:

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

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

ETC.SERVICES

This file supplies the information for these five services database functions:

- getservbyport()
- getservbyname()
- setservent()
- getservent()
- endservent()

The search order used to access this configuration file is:

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

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

Data Set Search Order Example

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

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//*********************************************************************
//*                                                                   *
//*                                                                   *
//*                                                                   *
//*         Descriptive Name:           FTP Server Start Procedure    *
//*                                                                   *
//*         File Name:                  tcpip.SEZAINST(EZAFTPAP)      *
//*                                     tcpip.SEZAINST(FTPD)          *
//*                                                                   *
```

```
//*         SMP/E Distribution Name:     EZAFTPAP                    *
//*                                                                   *
//*                                                                   *
//*             Licensed Materials - Property of IBM                  *
//*             This product contains "Restricted Materials of IBM"   *
//*             5694-A01 (C) Copyright IBM Corp. 1995, 2000.          *
//*             All rights reserved.                                  *
//*             US Government Users Restricted Rights -               *
//*             Use, duplication or disclosure restricted by          *
//*             GSA ADP Schedule Contract with IBM Corp.              *
//*             See IBM Copyright Instructions.                       *
//*                                                                   *
//*                                                                   *
//*********************************************************************
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM='POSIX(ON) ALL31(ON)/&PARMS'
.
.
.
//*      SYSTCPD explicitly identifies which file is to be
//*      used to obtain the parameters defined by TCPIP.DATA.
//*      The SYSTCPD DD statement should be placed in the JCL of
//*      the server.  The file can be any sequential data set,
//*      member of a partitioned data set (PDS), or HFS file.
//*SYSTCPD DD DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)

.
.
.
```

The search sequence for each configuration file is as follows:

- TCPIP.DATA
  1. Environment variable RESOLVER_CONFIG

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

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

     Because the SYSTCPD DD card is commented out in the above JCL, the search goes to the next step in the sequence. Using SYSTCPD for applications that fork() or use the exec functions to create child processes is not advisable because the child processes would not have access to the SYSTCPD DD card.
  4. *userid.*TCPIP.DATA, where *userid* is the user ID that is associated with the current security environment (address space or task/thread)
  5. SYS1.TCPPARMS(TCPDATA)

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

     TCPIP.TCPIP.DATA is searched last, if necessary.
- HOSTS.SITEINFO
  1. The value of the environment variable X_SITE

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

3. *userid*.HOSTS.SITEINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
4. *hlq*.HOSTS.SITEINFO

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

- HOSTS.ADDRINFO
  1. The value of the environment variable X_ADDR

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

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

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

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

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

MVS-Related Considerations

MVS System Symbols

Use of MVS system symbols in the PROFILE.TCPIP and OBEYFILE data sets is automatically supported. This automatic support first tries to use hiperspace memory files to perform the symbol translation, but if an error occurs, then a temporary HFS file will be used. The temporary HFS file is created in either the directory specified by the TMPDIR environment variable or, if the TMPDIR environment variable is not defined, in the /tmp directory.

For MVS system symbols in other configuration files, such as TCPIP.DATA, use the symbol translator utility, EZACFSM1, to translate the symbols before the files are read by TCP/IP. EZACFSM1 reads an input file and writes to an output file, translating any symbols in the process.

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

For more information about the use of MVS system services, refer to *z/OS MVS Initialization and Tuning Guide*.

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

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

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

Automatic Restart Manager (ARM)

Automatic restart manager is an MVS component that can automatically restart the TCP/IP stack after an abnormal end (ABEND).

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

```
REQUEST=REGISTER
ELEMENT=EZA<i>sysclonetcpname</i>
```

where:

- *sysclone* is a 1– or 2–character shorthand notation for the name of the MVS system. Refer to *z/OS MVS Initialization and Tuning Guide* for a complete description of the SYSCLONE static system symbol.
- *tcpname* is a 1– to 8–character name of the TCP/IP stack which registers with the automatic restart manager. For example, if the SYSCLONE value is 02 and the TCP/IP stack name is TCPCS, the resulting ELEMENT value is EZA02TCPCS.

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

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

Logging of System Messages

Syslog daemon (syslogd) is a server process that must be started as one of the first processes in your z/OS UNIX environment. TCP/IP server applications and components use syslogd for logging purposes and can also send trace information to syslogd. Servers on the local system use AF_UNIX sockets to communicate with syslogd; remote servers use AF_INET sockets. z/OS CS components use the **local1**, **daemon**, **mail**, **user**, and **auth facilities** names.

Note: Each application activates and deactivates traces in a slightly different manner. For details, refer to the chapter on the individual application in this book.

The syslog daemon reads and logs system messages to the MVS console, log files, SMF, other machines, or users as specified by the configuration file. If syslogd is not started, log data from some applications will be displayed on the MVS console. For more information on syslogd, refer to “Chapter 2. Customization” on page 63.

The syslogd facility uses a common mechanism for segregating messages. Table 3 shows the facilities used by z/OS CS functions which write messages to syslogd. The Primary Syslog Facility column shows the syslog facility used for most messages logged by the application. Some applications use other facilities for certain messages. Table 3 also shows any additional facilities.

Table 3. syslogd Facilities

Application	syslogd Record Identifications	Primary Syslog Facility	Other Syslog Facility
OTELNETD	telnetd	local1	auth
SENDMAIL	sendmail	mail	None
POPPER	popper	mail	None
ORSHD	rshd	daemon	auth
TCP/IP Configuration	Config	daemon	None
FTP Server	ftpd, ftps	daemon	None
Traffic Regulation Management Daemon (TRMD)	TRMD	daemon	None
ROUTED	routed	daemon	None
NAMED	named	daemon	None
Trap Forwarder Daemon	trapfwd	daemon	None

Table 3. syslogd Facilities (continued)

Application	syslogd Record Identifications	Primary Syslog Facility	Other Syslog Facility
OREXECD	rexecd	daemon	auth
Policy Agent (PAGENT)	Pagent	daemon	None
Service Level Agreement SNMP Subagent	PASubA	daemon	None
SNMP Agent (OSNMPD)	snmpagent	daemon	None
PWCHANGE Command	pwchange	daemon	None
PWTOKEY Command	pwtokey	daemon	None
syslogd	syslogd	daemon	None
DHCP Server	dhcpsd	user	None
TIMED Daemon	timed	user	None
TFTP Server	tftpd	user	None
OMPROUTE	omproute	user	None

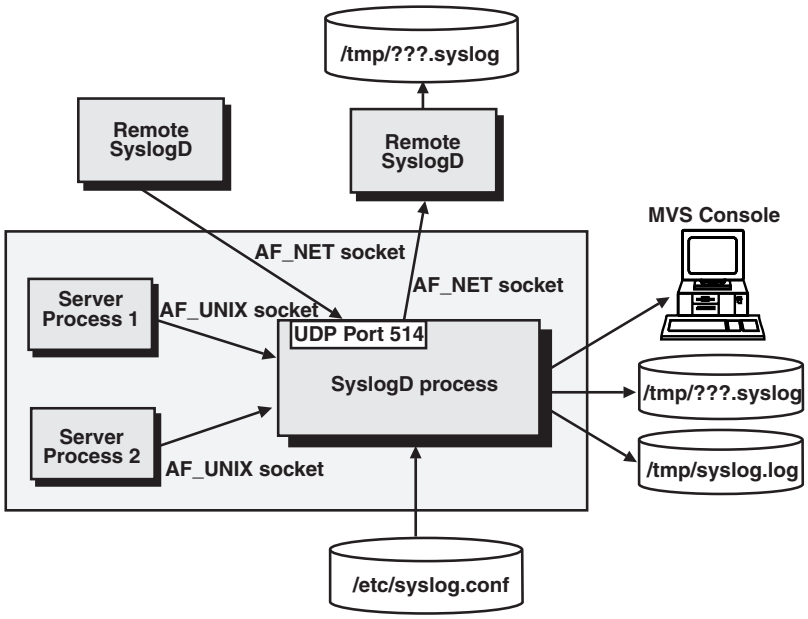


Figure 2. syslogd Operation

Note: /tmp/???.syslog. is the file specified in the syslogd.conf file.

Accounting - SMF Records

This section describes the system monitoring facility (SMF) records for the Telnet and FTP servers, API calls, syslogd, and FTP and Telnet client calls. The EZASMF76 macro can be used to map the TCP/IP SMF records. EZASMF76 produces assembler level DSECTs for the Telnet (Server and Client), FTP (Server and Client), and API SMF records.

Note: If the BPX.SMF facility is defined and SMF records are to be written by syslogd, the user ID with which syslogd runs must be permitted to BPX.SMF.

To create the Telnet SMF Record layout, code:

```
EZASMF76 TELNET=YES
```

To create the FTP SMF Record layout, code:

```
EZASMF76 FTP=YES
```

To create the API SMF Record layout, code:

```
EZASMF76 API=YES
```

SMF Accounting Issues

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

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

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

Security Considerations

The IBM Communications Server for z/OS, along with other elements of z/OS, provides numerous security technologies to protect mission-critical data in a TCP/IP environment. The following provides an overview of these z/OS technologies and how they can be used to secure an environment.

Cryptography

The foundation of good security methods begins with cryptography. Encryption services protect sensitive data from being read by other than the intended receiver. Cryptographic authentication and data integrity services allow communicating hosts to detect whether data is altered in transit. Public-key cryptography can identify and

authenticate hosts or users. Once a secure session is created, successful data authentication and decryption occurs only if both hosts have the correct session keys.

End-to-End Security

Cryptographic security solutions can be applied to a portion of the data path or end-to-end, whichever is appropriate for your security policy. Generally, the greatest degree of security is provided when cryptographic methods are used end-to-end. However, if only portions of the data path are considered untrusted by an enterprise (such as the Internet) it may be adequate to protect only the untrusted segment with cryptography. z/OS offers security protocols that can be configured to protect portions of the data path or the entire data path.

Workload-Based Security Deployment

In making a security protocol selection, an important consideration is the application workload to be protected. In order to illustrate this concept, it is helpful to understand where various protocols are implemented from a layering perspective.

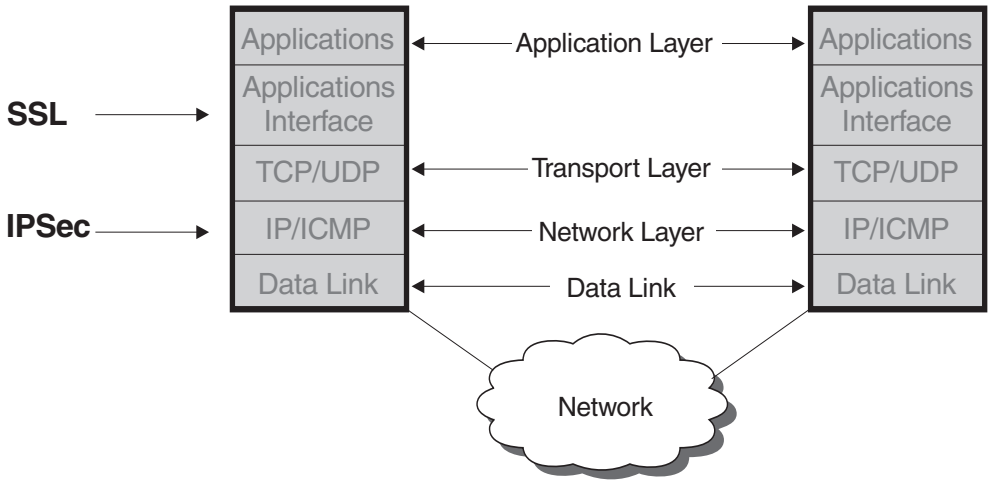


Figure 3. Security Protocol Selection

The network layer is the lowest layer in the protocol stack where end-to-end security over multiple hops can be applied. Network layer security protocols provide blanket protection for upper-layer application data without requiring modification to the application. IPSEC is implemented at the network layer and can be used to provide authentication, integrity, and data privacy between any two IP entities. IPSEC can protect a segment of the data path (for example, between two routers), or it can secure the data path end-to-end. Management of cryptographic keys and security associations can be manual or automated via an IETF defined key management protocol called the Internet Key Exchange (IKE).

IPSEC can protect selected traffic or all traffic. Selectivity can be based on various combinations of source and destination IP address, port, and protocol. This selectivity can be used to avoid the overhead of multiple security protocols when alternate security protocols are used to secure specific applications. For example, you might want to exclude web traffic (based on the well-known port of a web server) from IPSEC coverage because you would like to use SSL.

IPSEC allows the creation of Virtual Private Networks (VPN). A VPN enables an enterprise to extend its network across a public network such as the Internet through a secure tunnel (or security association). Figure 4 on page 29 shows some

of the typical IPSEC configurations. In Figure 4, IPSEC security associations are shown between two firewalls, between client and firewall, and between client and z/OS server.

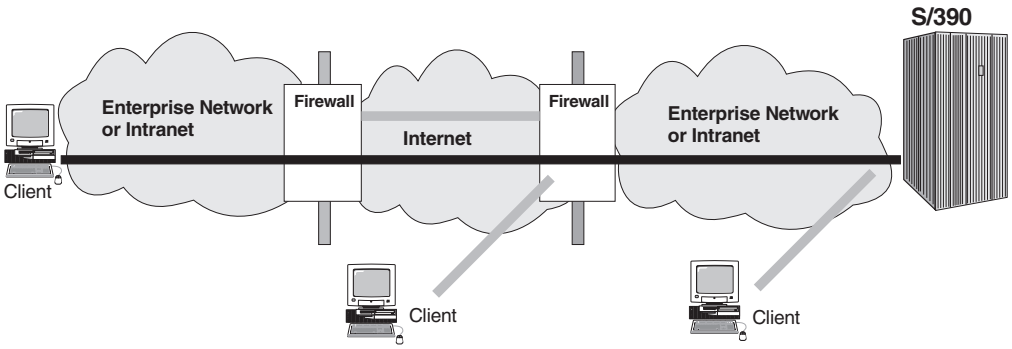


Figure 4. IPSEC Security Associations

Communications Server for z/OS IPSEC supports the latest RFCs including Triple DES for strong encryption. z/OS V1R1 supports IKE, which reduces the manual effort in managing and distributing IPSEC session keys and also supports a non-disruptive session key refresh capability. Ease of key management is a critical factor as the numbers of IPSEC hosts increase.

IPSEC is enabled in IBM Communications Server for z/OS by specifying the FIREWALL option on IPCONFIG in the PROFILE.TCPIP. See *z/OS Communications Server: IP Configuration Reference* for more information about the IPCONFIG statement. IPSEC is configured by using the z/OS Firewall Technologies product. For more information on z/OS Firewall Technologies and how to configure IPSEC in z/OS CS, refer to *z/OS SecureWay Security Server Firewall Technologies*.

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

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

Secure Sockets Layer (SSL) is another popular security protocol implemented above the network layer at the application interface layer. SSL, originally used to secure traffic between a web browser and web server, can also be used to secure other applications.

One such user of SSL is the TN3270 server provided by Communications Server for z/OS. Serving as a protocol gateway between the IP network and the SNA network, the SSL-enabled TN3270 server protects the data path in the IP network from the TN3270 client all the way to the S/390® TN3270 server. If the TN3270

Server resides on a different host from the target SNA application, SNA Session Level Encryption can be used to secure the SNA portion of the data path. In OS/390 V2R8, the TN3270 SSL support was augmented to support client authentication and pre-login access control using RACF® digital certificate support. Before users can receive a logon screen from TN3270, they must provide a certificate from a trusted certificate authority that is authenticated as part of the initial SSL session setup. Additionally, this support also verifies that the user associated with the certificate is authorized to access the TN3270 server port. See "Connection Security" on page 255 for more information. For new applications, security can be built-in. One method of building security into the application on S/390 is to use z/OS System SSL. Newer versions of network services such as SNMPv3 and Dynamic DNS have security built into the protocol using standards-based specifications for secure interoperability.

z/OS V1R1 Security Server shipped with Kerberos Version 5: z/OS CS V1R1 ships Kerberos Version 4. z/OS V1R1 Security Server ships a different Kerberos, Version 5. Because Security Server Kerberos does not require DCE login and eliminates the need for multiple registries, it is recommended that new applications be written to Kerberos Version 5 and use z/OS Security Server. Existing applications that use z/OS CS Kerberos can continue to use Kerberos Version 4. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about z/OS CS Kerberos.

UNIX System Services Security Considerations: This section describes some of the changes that have a product-wide effect. For descriptions of changes that affect specific servers or components, see the sections of this book that describe each server and component.

Requirement for an OMVS Segment: Many TCP/IP Services components in z/OS CS now exploit z/OS UNIX services in both the native MVS environment and in the z/OS UNIX environment. For example, all TCP/IP socket APIs (except the Pascal API) and TCP/IP applications (whether they are provided by z/OS CS, OS/390, other IBM and non-IBM products, or written by users) now make use of z/OS UNIX services.

Use of z/OS UNIX services requires an z/OS UNIX security context, referred to as an *OMVS segment*, for the user ID associated with any unit of work requesting these services. In other words, most user IDs requiring access to TCP/IP functions now require an OMVS segment to be defined in Resource Access Control Facility (RACF).

Note: The tasks, examples, and references in this section assume that you are using the z/OS CS Security Server (RACF). If you are using a security product from another vendor, read the documentation for that product for instructions on task performance.

Restrictions: None.

What This Change Affects::

- Security administration

Migration Procedures:

Table 4. Requirements of OMVS segment in RACF

Task	Details
Satisfy the requirement for an OMVS segment in RACF	Do one of the following: • Identify all the users in your environment that use TCP/IP services and then define OMVS RACF segments for the associated user IDs. • Use the default OMVS segment support provided by RACF and z/OS UNIX for users and groups.

The default OMVS segments reside in the USER profile and GROUP profile. The names of these profiles are identified by the installation, using the BPX.DEFAULT.USER facility class profile. The application data field in the class profile contains the user ID, or the user ID/group ID, that is used to access the default OMVS segments for users and groups, respectively.

Notes:

1. An HFS must be defined for the OMVS segment, and the home directory must exist.
2. If you use a trusted or privileged started task in ICHRIN03 or the STARTED class (especially a generic entry), be careful in assigning a default UID and GID with facility class BPX.DEFAULT.USER. Whenever trusted or privileged is specified, all default tasks have superuser authority.

To set up default OMVS segments, follow the steps in the following table.

Table 5. Setting up default of OMVS segment

Task	Details
Define a Group ID (GID) to the system to be used as an anchor for a default OMVS group segment.	Use the following command: `ADDGROUP OEDFLTG OMVS(GID(777777))` Make the GID unique so that it is easily identifiable. The GID can be either very high or very low. The other fields related to the GID are not likely to be used for anything.

Table 5. Setting up default of OMVS segment (continued)

Task	Details
Define a user ID (UID) to be used as an anchor for the default OMVS user segment.	Use the following commands: `ADDUSER OEDFLTU DFLTGRP(OEDFLTG) NAME('OE DEFAULT USER') OMVS(UID(999999) HOME('/') PROGRAM('/bin/sh'))` **Note:** To avoid giving superuser authority, do not use zero as the UID. When defining a UID, consider the following: • **UID** should be unique so that it is easily identifiable. The number can be very high or very low. • **HOME** — Use one of the following options when defining the home directory for the default user: – Define the HOME directory as the root (/). The users do not have write access. They do not need to update their home directory. – Define the HOME directory in the /tmp directory. – (*Not recommended*) Define a directory as you would for any other user. This directory is then used concurrently by many users that do not have an OMVS segment. • **PROGRAM** defines the default shell in this field. The other fields related to this UID are not likely to be used for anything.

Set up a default for the USER OMVS segment or set up a default UID and GID.	• To set up a default for the USER OMVS segment only, create a facility class profile named BPX.DEFAULT.USER, and then specify the default UID in the application data field. Use the following commands:
	`RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('OEDFLTU')`
	`SETROPTS RACLIST(FACILITY) REFRESH`
	
	Note: You cannot set up a default GROUP OMVS segment alone.
	• To set up a default UID and GID, create a facility class profile named BPX.DEFAULT.USER, and then specify the default UID and GID in the application data field. Use the following commands:
	`RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('OEDFLTU/OEDFLTG')`
	`SETROPTS RACLIST(FACILITY) REFRESH`
	
	Be aware that the facility class must be activated. In addition, the USER profile of the default UID and the GROUP profile of the default GID must exist, and must contain OMVS segments with a UID and GID, respectively.
	Note: RACF does not check to ensure that the application data points to a valid UID or UID and GID, or that the USER and GROUP profiles contain OMVS segments with the required UID and GID.

The following process shows how the BPX.DEFAULT.USER facility class profile works:

1. A user requests a UNIX service, which is serviced by the kernel.
2. The kernel calls the security product to extract the UID, GID, HOME, and PROGRAM information.

3. The security product attempts to extract the OMVS segment associated with the user. If the user is not defined, the security product attempts to extract and use the OMVS segment for the default user that was listed in the BPX.DEFAULT.USER profile.

A similar process is followed to obtain a GID when the user default group does not have an OMVS segment.

Authorization of TCP/IP Started Task User ID: The AF_INET physical file system of z/OS UNIX uses the TCP/IP system address space as an AF_INET transport provider. For this to occur, the TCP/IP system address space must connect to z/OS UNIX and become an z/OS UNIX process. Therefore, the started task UID that is assigned to the TCP/IP system address space must have a valid OMVS segment.

As an AF_INET transport provider, the TCP/IP system address space requires superuser privileges in z/OS UNIX.

Table 6. Requirements of Superuser Privileges in z/OS UNIX

Task	Details
Define the TCP/IP system address space started task UID as UID=0 or else define the TCP/IP system address space as a trusted environment in the RACF started class profile for the TCP/IP system address space.	Use the following command to assign an OMVS segment to the TCP/IP started task user ID specified as UID=0: `ALU tcpip_userid OMVS(UID(0) HOME(/) PGM(/bin/sh))`

Other User IDs Requiring z/OS UNIX Superuser Authority: When a started procedure is used to start the following servers, daemons, and agents, the user must be a superuser [UID(0)] or permitted to BPX.SUPERUSER class profile.

- File transfer protocol (FTP) daemon
- Domain name system (DNS) server
- OROUTED server
- SNMP agent (OSNMPD)

The following daemons are managed by the Inetd Generic Listener Program, and the user specified in the inetd.conf file in the /etc directory must have a UID(0) specified in the inetd.conf file in the /etc directory. For details on Inetd, refer to *z/OS UNIX System Services Planning*. For details on individual daemons, refer to the *z/OS Communications Server: IP Configuration Reference*.

- z/OS UNIX remote execution daemon (REXECD)
- z/OS UNIX remote shell daemon (RSHD)
- z/OS UNIX TELNET daemon

BPX.DAEMON Facility Class: Certain z/OS CS TCP/IP Services servers need to change the security environment of the process in which they currently execute. For example, the FTPD daemon creates a new z/OS UNIX process for every FTP client connecting to it. After the new process is created, the daemon changes the security environment of the process so that it is associated with the security context of the FTP client. The RACF facility class resource BPX.DAEMON is used for this purpose.

Table 7. BPX.DAEMON

Task	Details
Decide if you want to activate the BPX.DAEMON level of security by reviewing the section about BPX.DAEMON authority in *z/OS UNIX System Services Planning* to determine whether this level of security is appropriate for your installation.	This is not required. It is recommended, however, because it provides additional security in the z/OS UNIX environment. The following TCP/IP Services servers and daemons in z/OS CS change the security environment of their processes: • z/OS UNIX TELNETD • z/OS UNIX RSHD • z/OS UNIX REXECD • FTPD
Plan the time at which you define BPX.DAEMON carefully.	As soon as you define the BPX.DAEMON resource, MVS will not let programs change the security environment unless the programs are retrieved from a program-controlled library and unless the UID under which the program executes has access to BPX.DAEMON.
If you decide **not** to define the BPX.DAEMON facility class, assign UID(0) for the UIDs associated with these servers and daemons.	This is sufficient for processing. It is described in "Other User IDs Requiring z/OS UNIX Superuser Authority" on page 33.
If you decide **to** define the BPX.DAEMON facility class, grant READ access to this profile for the UIDs associated with the listed daemons. Also, enable BPX.DAEMON security by defining the BPX.DAEMON facility class profile in RACF	To define the BPX.DAEMON facility class profile in RACF, use the following command: `RDEFINE FACILITY BPX.DAEMON UACC(NONE)` **Note:** You must specify the name BPX.DAEMON in this command. Substitutions for the name are not allowed.

If all the required conditions are not met, your server programs will fail as soon as you define BPX.DAEMON. If the server programs fail, delete BPX.DAEMON, and the setup reverts to its previous state. Check all your definitions, and make the required corrections before trying to define BPX.DAEMON again.

If this is the first facility class profile that your installation is using, activate the facility class using the following commands:

```
SETROPTS CLASSACT(FACILITY) GENERIC(FACILITY) AUDIT(FACILITY)
SETROPTS RACLIST(FACILITY)
```

If you start server programs using MVS start commands or from shell scripts that execute after startup of z/OS UNIX, you must allow the UIDs access to the BPX.DAEMON facility class resource. The following example shows the UID (ftpd_user_ID) with which you can start the FTPD daemon:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(ftpd_user_ID) ACCESS(READ)
```

Authorization to change the user security environment is granted only if both of the following two conditions are true:

- The server program is executing under a UID that has READ permission to the BPX.DAEMON facility class profile and a UID=0.
- All programs running in the address space have been retrieved from a controlled library. Program control is discussed in the following section.

Program Control: In an z/OS UNIX environment, there are additional security concerns related to the HFS and the loading of programs that are considered trusted. Program control facilities in RACF and z/OS UNIX provide a mechanism for ensuring that the z/OS UNIX program loading process has the same security features that APF authorization provides in the native MVS environment.

It is recommended that you enable program control in your installation. If you define the BPX.DAEMON facility class, then you **must** enable program control for certain z/OS CS load libraries. Review the section on program control in *z/OS UNIX System Services Planning* to decide whether program control is appropriate for your installation.

To enable program control, follow the tasks in the following table.

Table 8. Program Control

Task	Details
Activate program control.	Use the following command: `SETROPTS WHEN(PROGRAM)`

Set the universal access for public library data sets (those in LINKLST*xx*) to READ. This allows access to the controlled programs and any other program in those libraries. (MVS opens the LNKLST*xx* libraries during IPL and makes these programs public. However, users cannot make changes.)	Use the following commands to create RACF data set profiles:
	`ADDSD 'cee.version.SCEERUN' UACC(READ)`
	`ADDSD 'SYS1.LINKLIB' UACC(READ)`
	`ADDSD 'TCPIP.SEZALINK' UACC(READ)`
	`ADDSD 'TCPIP.SEZATCP' UACC(READ)`

Ensure all load modules that are loaded by the BPX.DAEMON servers into an address space come from controlled libraries.	If the MVS contents supervisor loads a module from a noncontrolled library, the address space becomes ″dirty″ and loses its authorization. To prevent this from happening, define all the libraries from which load modules can be loaded as program controlled. At a minimum, this should include the C run-time library, the TCP/IP Services SEZALINK and SEZATCP libraries, SYS1.LINKLIB, and any load libraries containing FTP security exits.
	
	Use the following commands:
	`RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'/volser/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM('cee.version.SCEERUN'/volser/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM('TCPIP.SEZALINK'/volser/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM('TCPIP.SEZATCP'/volser/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM(db2.DSNLOAD'/'volser'/NOPADCHK UACC(READ)`
	`RALTER PROGRAM * ADDMEM(db2.DSNEXIT'/'volser'/NPPADCHK UACC(READ)`
	`RALTER PROGRAM * ADDMEM('ftp.userexits'/'volser'/NOPADCHK UACC(READ)`
	
	Note: If you define the load libraries as controlled, do not specify a universal access of NONE for the PROGRAM resources. If you do so for your SYS1.LINKLIB programs, you cannot IPL your MVS system. Be aware also that in OS/390, the volser specification is optional.

Activate RACF changes.	Use the following command: `SETROPTS WHEN(PROGRAM) REFRESH`

z/OS CS relies on a Security Authorization Facility (SAF) to protect several resources:

- Started tasks require access to a STARTED resource. This is documented in the server information in the *z/OS Communications Server: IP Configuration*

Reference. Also, refer to SEZAINST(EZARACF) for SAF authorizations required for the TCP/IP stack and servers started tasks.

- Restricting access to a network is provided by SERVAUTH resource. z/OS CS provides a one-to-one mapping between network and SAF resource names such that users not permitted access to the SAF resource for a network are not allowed to communicate with the network. Refer to the NETACCESS statement in *z/OS Communications Server: IP Configuration Reference* or "Setting Up SAF Server Access Authorization (SERVAUTH) Optional" on page 98 for more information.

  Restricting access to port numbers by applications is also provided by SERVAUTH. z/OS CS provides a one-to-one mapping between port numbers and SAF resource names. Refer to the PORTACCESS statement in the*z/OS Communications Server: IP Configuration Reference* or "Setting Up SAF Server Access Authorization (SERVAUTH) Optional" on page 98 for more information.

  Also, similar to PORTACCESS, z/OS CS ensures a user attempting to connect to a TN3270 secure port is allowed access to the port. This support is used in conjunction with TN3270 SSL client authentication support. Refer to the CLIENTAUTH statement in the *z/OS Communications Server: IP Configuration Reference* or "Setting Up SAF Server Access Authorization (SERVAUTH) Optional" on page 98 for more information.

  Restricting access to the TCPIP stack is also controlled under z/OS CS by the SERVAUTH resource. Refer to "Setting Up SAF Server Access Authorization (SERVAUTH) Optional" on page 98 for more information.
- Restricting access to operator commands is provided through the OPERCMDS resource. z/OS CS verifies that users have access to specific OPERCMDS resources before executing the operator command. Refer to the operator commands information in the *z/OS Communications Server: IP Configuration Reference* or "Setting Up SAF Server Access Authorization (SERVAUTH) Optional" on page 98 for more information about limiting access to z/OS CS commands.

Defining TCP/IP as a UNIX System Services Physical File System (PFS)

As described in *z/OS Communications Server: IP Migration*, the TCP/IP Services stack in z/OS CS must be defined as an z/OS CS UNIX System Services PFS before it can be started. This involves updating the BPXPRM*xx* parmlib member. The following sample definitions in BPXPRM*xx* define TCP/IP as an z/OS CS UNIX System Services PFS:

```
FILESYSTYPE TYPE(INET)  ENTRYPOINT(EZBPFINI)

NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(60000)
        TYPE(INET)
```

The BPXPRM*xx* member contains additional z/OS CS UNIX System Services parameters that are crucial to the proper operation of TCP/IP. Carefully examine and specify these parameters:

- MAXPROCSYS — Specifies the maximum number of OS/390 UNIX processes that the system allows.
- MAXPROCUSER — Specifies the maximum number of processes associated with a single z/OS CS UNIX System Services user ID.
- MAXUIDS — Specifies the maximum number of OS/390 UNIX user ids that can operate concurrently.

- MAXFILEPROC — Specifies the maximum number of z/OS CS UNIX System Services file descriptors an z/OS CS UNIX System Services process can allocate. This includes access to both HFS files and z/OS CS UNIX System Services socket descriptors. In z/OS CS, most TCP/IP applications access z/OS CS UNIX System Services sockets, either directly or indirectly, using the TCP/IP socket APIs. You should set the MAXFILEPROC value high enough to accommodate the largest number of sockets a single TCP/IP application (or z/OS CS UNIX System Services process) can allocate.

  Be aware that the tn3270 Telnet server is exempt from the limit specified in this parameter. The tn3270 Telnet server can obtain the maximum number of socket connections for a single z/OS CS UNIX System Services process.
- MAXPTYS — Specifies the maximum number of pseudo-terminals for the system.
- MAXTHREADTASKS — Specifies the maximum number of MVS tasks that a single process can have concurrently active.
- MAXTHREADS — Specifies the maximum number of threads that a single process can have concurrently active.
- MAXQUEUEDSIGS — The sum of MAXQUEUEDSIGS and MAXFILEPROC multiplied by 2 is the system wide maximum for the total number of asynchronous OS/390 UNIX socket calls that can be outstanding. When specifying this number, consider the following:
  - For every TCP/IP connection that the TN3270 Telnet Server has, there is an asynchronous OS/390 UNIX socket call outstanding. This is true for both TN3270 and TN3270E clients.
  - Any TCP/IP application, IBM or vendor supplied, that uses either the OS/390 UNIX Assembler Callable Services asyncio call or the TCPIP provided Sockets Extended asynchronous API could have one or more outstanding asynchronous socket calls.

The MAXSOCKETS() parameter specifies the total number of z/OS CS UNIX System Services sockets that can be active at any one time. You must ensure that this specification is large enough to accommodate your installation's workload. For example, each connection to your tn3270 Telnet server or FTP server requires one z/OS CS UNIX System Services socket. Once the maximum number of sockets is allocated, then no more Telnet sessions, FTP sessions, or other applications that require z/OS CS UNIX System Services sockets can be started.

References

For details on the BPXPRM*xx* member, please refer to the following guides:

- *z/OS UNIX System Services Planning*
- *z/OS MVS Initialization and Tuning Reference*
- *z/OS UNIX System Services File System Interface Reference*

Considerations for Multiple Instances of TCP/IP

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

- Increased capacity and availability

  In a multiple-processor system, multiple TCP/IP stacks would be configured to take advantage of the multiple processors.
- Isolation of z/OS UNIX applications from TCP/IP applications

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

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

In addition, running multiple z/OS Communications Server TCP/IP stacks requires additional system resources, such as storage, CPU cycles, and DASD. It also adds a significant level of complexity to the system administration tasks for TCP/IP.

For help in configuring multiple instances of TCP/IP, refer to the z/OS UNIX manuals. In particular, refer to information about setting up for common sockets in *z/OS UNIX System Services Planning*.

AF_INET and Common INET Physical File System (CINET PFS)

When z/OS UNIX processes an API request such as socket() or read(), the logical file system (LFS) determines what PFS will handle the request. For example, a read() request can be used with a socket or with a file identifier (FID). The LFS is used to determine which PFS is called.

The following are examples of PFS's in an z/OS UNIX environment:

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

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

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

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

Port Management Overview

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

Port management becomes more complex in an environment where there are multiple transport providers (multiple instances of TCP/IP) and a potential for multiple combinations of the same server (for example, z/OS UNIX and TN3270/TN3270E Telnet).

In a multiple transport provider environment, the following questions need to be answered for each server in an installation:

- Is the server generic so that it can communicate with multiple TCP/IPs or does the server have an affinity for one instance of the transport providers and can only communicate with one TCP/IP?
- How can ports be reserved across multiple transport providers? When is the port reservation determined by MVS rather than by the job name, procedure name, or user ID?
- How can you synchronize between BPXPARMS and PORTRANGE for ephemeral port reservation?
- How can TCP/IP distinguish between two different instances of Telnet (z/OS UNIX TELNET and TN3270/TN3270E Telnet)?

Generic Server Versus Server with Affinity for a Specific Transport Provider

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

Generic Server: A generic server, a server without an affinity for a specific transport provider, provides service to any client on the network. (See Figure 5.) FTP is an example of a generic server. The transport provider is merely a connection linking client and server. The service File Transfer is not related to the internal functioning of the transport provider, and the server can communicate concurrently over any number of transport providers.

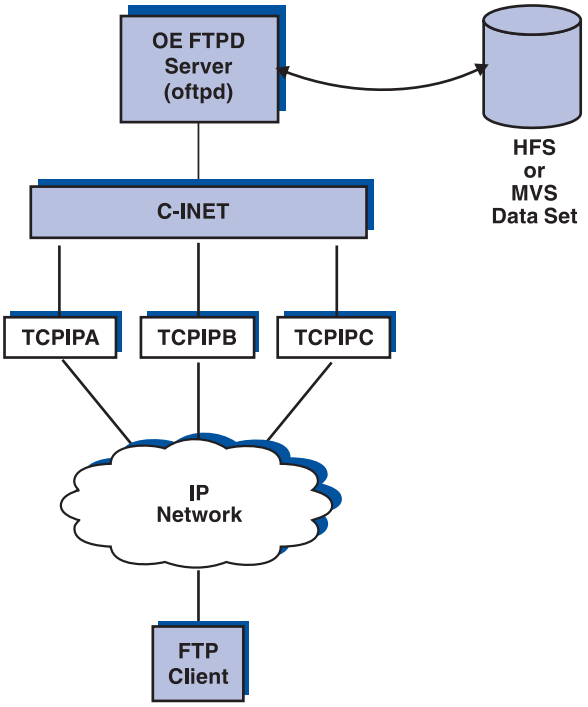


Figure 5. Generic Server

Server with an Affinity for a Specific Transport Provider: When the service is related to the internal functioning of the transport provider (for example, TELNET, OMPROUTE, OSNMPD, and onetstat, a command), there must be an explicit

binding of the server application to the chosen transport provider. (See Figure 6 .) There must also be a way to specify the single transport to be chosen.

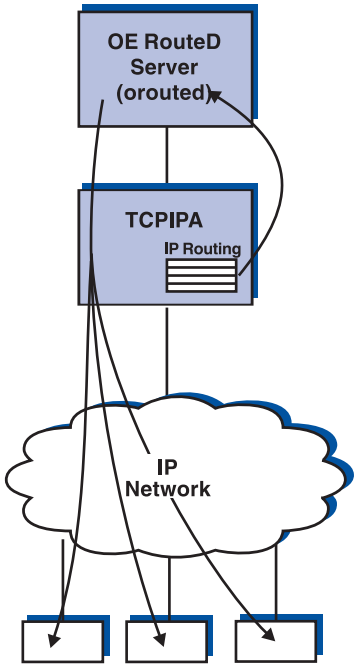


Figure 6. Server with Affinity for a Specific Transport Provider

With the exception of applications that use the socket API provided by TCP/IP, other IBM-supplied applications that use the z/OS UNIX socket API and that must bind to a specific transport provider use the z/OS UNIX socket call `setibmopt()` (refer to *z/OS C/C++ Run-Time Library Reference*) to specify which TCP they have chosen. A C function `__iptcpn()`, described in the *z/OS C/C++ Run-Time Library Reference*, enables the application to search the TCPIP.DATA file to find the name of the specific TCP/IP. (See Figure 7.)

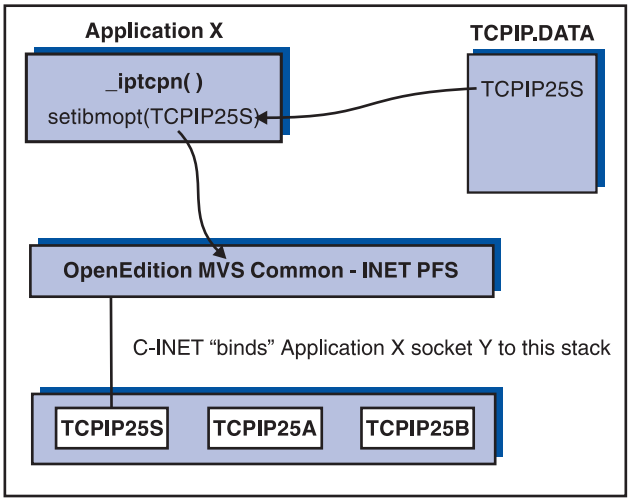


Figure 7. Example of Binding an Application to a Specific Transport Provider

Port Number Conflicts in a C-INET Environment

In z/OS CS, you can configure multiple TCP/IP stacks in a single MVS image using the C-INET feature. In a C-INET configuration, an application using the z/OS UNIX socket interface can get transparent access to all the TCP/IP protocol stacks configured under C-INET. For example, when an application coded to z/OS UNIX sockets performs a SOCKET/BIND/LISTEN in a C-INET environment, the request is propagated by C-INET to all the TCP/IP stacks. This application can then service client requests that arrive into any of the configured TCP/IP stacks without having any awareness of this fact. This type of application is often referred to as a ″generic server/daemon″.

The following servers/daemons shipped by z/OS CS are generic servers/daemons:

- FTPD
- z/OS UNIX RSHD
- z/OS UNIX REXECD
- z/OS UNIX TELNETD
- z/OS UNIX SENDMAIL
- z/OS UNIX POPPER
- TFTPD
- TIMED
- z/OS UNIX Portmap

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

If a server started by INETD (a generic server) requires affinity to a specific stack, this affinity can be accomplished by use of the _BPXK_SETIBMOPT_TRANSPORT environment variable. For more information about the _BPXK_SETIBMOPT_TRANSPORT environment variable refer to *z/OS UNIX System Services Planning*.

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

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

```
//FTPD   PROC MODULE='FTPD',PARMS='TRACE'
//FTPD   EXEC PGM=&MODULE,REGION=7M,TIME=NOLIMIT,
//       PARM=('POSIX(ON) ALL31(ON)',
//      'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE")',
//      '/&PARMS')
//CEEDUMP  DD SYSOUT=*
//*
//*       SYSFTPD is used to specify the FTP.DATA file for the FTP
```

```
//*       server.  The file can be any sequential data set, member
//*       of a partitioned data set (PDS), or HFS file.
//*
//*       The SYSFTPD DD statement is optional.  The search order for
//*       FTP.DATA is:
//*
//*           /etc/ftp.data
//*           SYSFTPD DD statement
//*           jobname.FTP.DATA
//*           SYS1.TCPPARMS(FTPDATA)
//*           tcpip.FTP.DATA
//*
//*       If no FTP.DATA file is found, FTP default values are used.
//*SYSFTPD DD DISP=SHR,DSN=TCPIP.SEZAINST(FTPSDATA)
//*
//*      SYSTCPD explicitly identifies which file is to be
//*      used to obtain the parameters defined by TCPIP.DATA.
//*      The SYSTCPD DD statement should be placed in the JCL of
//*      the server.  The file can be any sequential data set,
//*      member of a partitioned data set (PDS), or HFS file.
//SYSTCPD DD DISP=SHR,DSN=SYS1.TCPPARMS(TCPDATA)
//*
//*      SYSFTSX explicitly identifies which file is to be used
//*      for the EBCDIC-ASCII translation table.  The file can
//*      be any sequential data set, member of a partitioned data
//*      set (PDS), or HFS file.
//SYSFTSX DD DISP=SHR,DSN=TCPV34.STANDARD.TCPXLBIN
```

All the parameters specified prior to the slash (/) in the parameter statement are processed by the C/C++ run time library. Parameters to be passed to the FTPD program must appear after the slash (/). Also note how the parameters were split over three lines in this example because they could not fit on a single line.

The following example uses JCL for the started procedure for INETD:

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

The STDENV data set would contain the _BPX_SETIBMOPT_TRANSPORT variable as follows:

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE
```

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

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

Notes:

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

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

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

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

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

   ```
   EDC5112I Resource temporarily unavailable.
   ```
3. For more detailed information about requesting transport affinity, refer to *z/OS UNIX System Services Planning*.

Port Reservation across Multiple Transport Providers

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

For more information about reserving ports with the PORT statement, see "Chapter 2. Customization" on page 63.

Ephemeral Ports: When running with multiple transport providers, just as it is necessary to synchronize PORT reservations for specific applications across all stacks, it is required to synchronize reservations for port numbers that will be dynamically assigned across all stacks. These are the ephemeral ports above 1023, which are assigned by the stack when none is specified on the application `bind()`. To reserve a group of ports in the PROFILE.TCPIP, use PORTRANGE. For more information about PORTRANGE, see "Chapter 2. Customization" on page 63. Specify the same PORTRANGE for every stack. In addition, you need to let the z/OS UNIX C-INET know which ports are guaranteed to be available on every stack. The following is an example of reserving ports 4000 to 4999 in the two required files:

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

Selecting a Stack When Running Multiple Instances of TCP/IP

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

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

In order to answer these questions, a distinction must be made between standard servers and clients (those that come with the z/OS CS product), and other socket application programs, including those you might have written yourself.

Standard Servers and Clients

The anchor configuration data set is the TCPIP.DATA data set. This is the main resolver configuration data set with information on host name, domain origin, and so on. It holds the TCPIPJOBNAME parameter, which identifies the TCP/IP stack to use, and the DATASETPREFIX parameter, which is used by the resolver code and other services when allocating configuration data sets (HOSTS.SITEINFO, HOSTS.ADDRINFO, ETC.SERVICES, ETC.PROTO, and STANDARD.TCPXLBIN). See “ETC.PROTO” on page 21 and “ETC.SERVICES” on page 21 for more information about ETC.PROTO and ETC.SERVICES.

The key to selecting both a specific stack and resolver configuration data sets is to control which TCPIP.DATA data set a standard server or client address space allocates. Applications that use the UNIX API may use Common INET to determine which stack an application will use. But, it is important to ensure that the following search order and the contents of the resolver configuration data set are understood.

Standard servers and clients search for TCPIP.DATA in the following sequences as described in “TCPIP.DATA Search Order” on page 14:

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

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

1. Environment variable ″RESOLVER_CONFIG=file/dataset″
2. /etc/resolv.conf
3. //SYSTCPD DD

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

Nonstandard Servers and Clients

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

- The socket API used to create the program
- The release of z/OS CS they use

If you run sockets programs from other products or vendors, you may want to know which sockets API was used to develop the program, and which techniques, if any, the program uses to specify the name of the TCP/IP system address space. As of z/OS CS V2 R10, this task is made easier than with previous releases. As long as application programs that use an TCP/IP socket library do not specify anything specific on calls `setibmopt()` or `INITAPI`, the TCPIPJOBNAME from a TCPIP.DATA data set will be used as the last resort for finding a TCP/IP system address space name.

Table 9 depicts the differences that prevail in stack selection depending on the socket API under which you are running the socket program.

Table 9. How Your Own Socket Programs Select a Stack

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

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

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

In an z/OS UNIX environment, the TCP/IP system address space is not selected by the socket application program, but rather by the AF_INET PFS. In the z/OS UNIX Common INET (C-INET) environment, your application will be associated with multiple TCP/IP stacks unless the application specifically associates with a particular stack via the z/OS UNIX socket call setibmopt().

TCP/IP TSO Clients

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

- Add a SYSTCPD DD statement to your TSO logon procedure. The issue with this approach is that a separate TSO logon procedure per stack is required, and users have to log off TSO and log on again using another TSO logon procedure in order to switch from one stack to another.

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

The last method can be implemented by creating a small REXX program for every TCP/IP stack on your MVS system. For each stack create a REXX program with the name of the stack (for example, T18A or T18B). Whenever TSO users want to use the T18A stack, they run the T18A REXX program. Any TCP/IP functions invoked thereafter will use the T18A stack for socket communication. If users want to switch to the T18B stack, they run the T18B REXX program. See Figure 8 for an example.

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

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

exit(0)
```

Figure 8. REXX Program to Switch TSO User to Another TCP/IP Stack

Selecting Configuration Data Sets

The resolver code and other services that execute as part of the socket program address space to service calls such as `gethostbyname()`, `getservbyname()` and `getprotobyname()` allocate one or more configuration data sets to service these calls. All socket programs, including standard servers and clients and homegrown socket programs, need access to TCPIP.DATA, HOSTS.ADDRINFO, HOSTS.SITEINFO, ETC.SERVICES, ETC.PROTO and STANDARD.TCPXLBIN. (See Figure 9 on page 47 for more information regarding selecting configuration data sets.)

- **MVS**

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

  /etc/resolv.conf and /etc/hosts

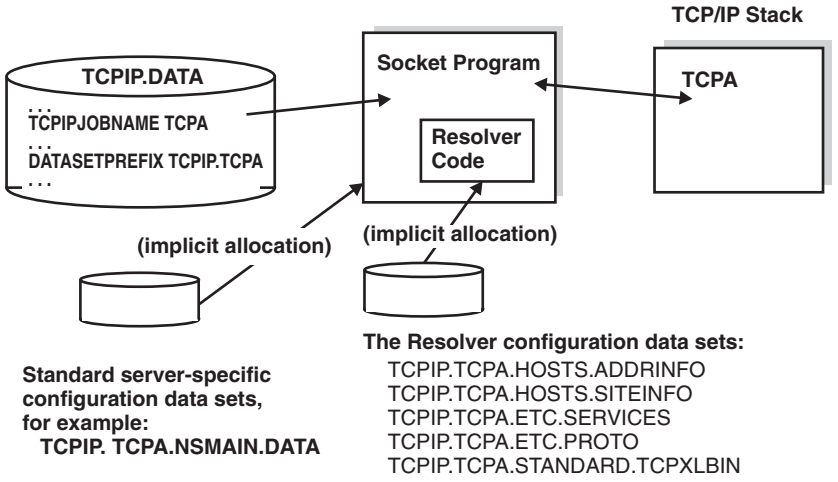


Figure 9. Selecting Configuration Data Sets

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

Sharing Resolver Configuration Data Sets

The general recommendation is to use separate DATASETPREFIX values for each stack and create separate copies of the required configuration data sets; at the very least, create separate copies of the resolver configuration data sets. For a test and a production stack, however, you would probably use different DATASETPREFIX values. However, if the stacks are functionally identical, you may share the same DATASETPREFIX values and many of the same configuration data sets. You need separate TCPIP.DATA data sets because of the two different TCPIPJOBNAMEs. On the other hand, you may choose to share the resolver configuration data sets between the stacks by using the same DATASETPREFIX value in each TCPIP.DATA data set. (See Figure 10 on page 48.)

The following example shows BPXPARMS that are needed for z/OS .

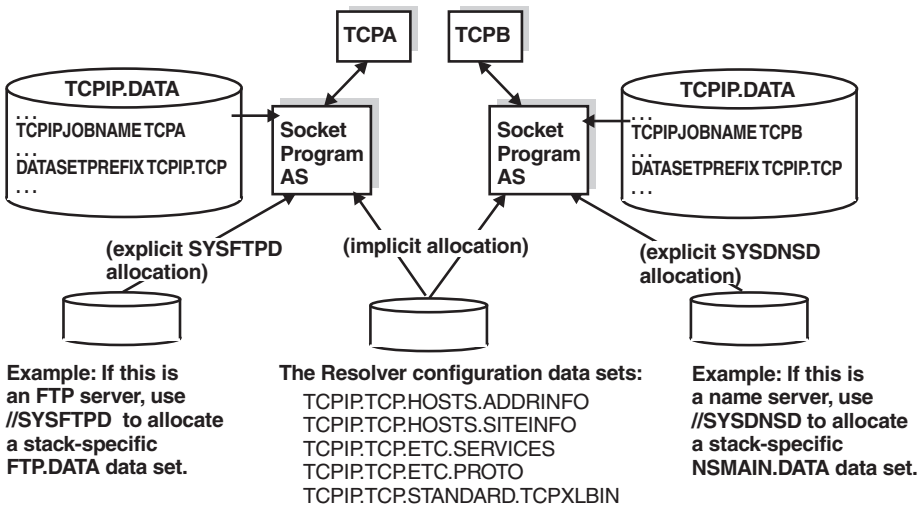


Figure 10. Sharing DATASETPREFIX

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

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

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

Specifying BPXPRMxx Values for a C-INET Configuration

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

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

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

Figure 11. SYS1.PARMLIB(BPXPRMxx) for Converged Sockets

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

2 The MAXSOCKETS operand specifies the maximum number of sockets that can be obtained for the given file system type. It should be large enough for the number of sockets needed for applications using z/OS CS.

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

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

5 EZBPFINI identifies a z/OS CS TCP/IP stack.

6 Keyword DEFAULT specifies which transport provider stack is to be used as the default stack for z/OS UNIX. If DEFAULT is not specified, the first stack will be used as the default stack. The sequence of SUBFILESYSTYPE statements is arbitrary if one stack is identified with the keyword DEFAULT. TCPIP1A is the default stack in Figure 11.

Considerations for Enterprise Extender

The Enterprise Extender (EE) network connection is a simple set of extensions to the existing open high-performance routing (HPR) technology. It performs an efficient integration of the HPR frames using UDP/IP packets. To the HPR network, the IP backbone is a logical link. To the IP network, the SNA traffic is UDP datagrams that are routed without any hardware or software changes to the IP backbone. Unlike gateways, there is no protocol transformation and unlike common tunneling mechanisms, the integration is performed at the routing layers without the overhead of additional transport functions. The advanced technology enables efficient use of the intranet infrastructure for support of IP-based client accessing

SNA-based data (for example, TN3270 emulators or Web browsers using services such as IBM's Host On-Demand) as well as SNA clients using any of the SNA LU types.

Enterprise Extender seamlessly routes packets through the network protocol *edges*, eliminating the need to perform costly protocol translation and the store-and-forward associated with transport-layer functions. Unlike Data Link Switching (DLSw), for example, there are no TCP retransmit buffers and timers and no congestion control logic in the router because it uses connectionless UDP and the congestion control is provided end system to end system. Because of these savings, the *edge* routers have less work to do and can perform the job they do best, which is forwarding packets instead of incurring protocol translation overhead and maintaining many TCP connections. Data center routers can handle larger networks and larger volumes of network traffic, thus providing more capacity. For more information, refer to the EE information in *SNA and TCP/IP Integration (SG24–5291–00)* (an IBM Redbook) or the website located at the following URL:

http://www-4.ibm.com/software/network/commserver/library/whitepapers/csos390.html.

Considerations for VIPA

The Internet Protocol is a connectionless protocol. IP packets are routed from the originator through a network of routers to the destination. All physical adapter devices in such a network, including those for client and server hosts, are identified by an IP Address which is unique within the network. The important point about IP is that a failure of an intermediate router node or adapter will not prevent a packet from moving from source to destination, as long as there is an alternate path through the network.

TCP sets up a connection between two endpoints, identified by the respective IP addresses and a port number on each. Unlike failures of an adapter in an intermediate node, if one of the endpoint adapters (or the link leading to it) fails, all connections through that adapter fail and must be reestablished. If the failure is on a client workstation host, only the relatively few client connections are disrupted and usually only one person is inconvenienced. However, an adapter failure on a server means that hundreds or thousands of connections may be disrupted. On an S/390 with large capacity, the number may run to tens of thousands.

A Virtual IP Address, or VIPA in TCP/IP for z/OS , alleviates this situation. A VIPA is configured in the same way as a normal IP address for a physical adapter, except that it is not associated with any particular device. To an attached router, the TCP on z/OS simply looks like another router. When the TCP receives a packet destined for one of its VIPAs, the inbound IP function of the stack notes that the IP address of the packet is in the stack's Home list and passes the packet up the stack. Assuming the stack has multiple adapters or paths to it (including XCF from other TCP stacks in a sysplex), if a particular physical adapter fails, the attached routing network will simply route VIPA-targeted packets to the stack via an alternate route.

While this removes hardware and associated transmission media as a single point of failure for large numbers of connections, the connectivity of a server can still be lost through a failure of a single stack or an MVS image. The VIPA can be configured on another stack with a manual process, but this requires the presence of an operator or programmed automation.

Dynamic VIPA Takeover enables Dynamic VIPAs to be moved without human intervention or programmed automation to allow new connections to a server at the same IP address as soon as possible. This can reduce downtime significantly. With Dynamic VIPA Takeover you can configure one or more TCP/IP stacks to be backups (VIPABACKUP statement) for a particular Dynamic VIPA. If the stack or MVS image where the Dynamic VIPA is active fails, one of the backup stacks automatically activates that Dynamic VIPA. The existing connections will be terminated but can be quickly reestablished on the taking over stack.

Note: Because a VIPA is associated with an z/OS TCP/IP stack and is not associated with a specific physical network attachment, it can be moved to a stack on any image in the sysplex or even to an z/OS TCP/IP stack not in the sysplex as long as the address fits into the installations network configuration.

You may also associate a particular Dynamic VIPA address with an application using the IOCTL SIOCSVIPA command or by BINDing explicitly to the Dynamic VIPA address. If the Dynamic VIPA address is within the VIPARANGE profile statement then this Dynamic VIPA address will be created dynamically. This type of configuration enables a Dynamic VIPA to become an address of an application in a sysplex.

With Sysplex Distributor you can spread connection requests destined for Dynamic VIPAs to other stacks in the sysplex. You can use the VIPADISTRIBUTE profile statement to designate up to 32 stacks where connections for a particular DVIPA and up to four ports, can be distributed, including the stack where the DVIPA is defined. The distributing stack (the stack where the VIPADISTRIBUTE statement was coded) might use either WLM or a combination of WLM and Quality of Service (QOS) performance information to determine where to forward new connection requests. If the distributing stack/MVS image fails, connections forwarded to target stacks can be preserved by having the Dynamic VIPA address backed up on another stack.

Similarly, a stack can immediately take back a Dynamic VIPA address from another stack. If the original stack VIPADEFINEd the address with the keyword MOVEABLE IMMEDIATE (the default), then the Dynamic VIPA is moved as soon as the second stack requests ownership. The second stack assumes responsibility for forwarding packets for existing connections to the appropriate stack. If MOVEABLE WHENIDLE was specified, ownership does not pass until all existing connections on the current stack are closed.

For detailed information about VIPA, see “Chapter 3. Virtual IP Addressing” on page 109.

Required Steps Before Starting TCP/IP

The following sections describe the steps you must complete before starting TCP/IP.

Planning Your Installation and Migration

It will be to your advantage to have studied thoroughly the following documentation prior to the installation and customization of IBM z/OS :

- Program Directory for z/OS for CBPDO Installation and ServerPac Reference, Program Number 5647-A01
- Preventive Service Planning (PSP) bucket
- The z/OS Web pages: http://www.s390.ibm.com/os390/installation/

- *z/OS Communications Server: IP Migration*
- *z/OS UNIX System Services Planning*
- OS390CKL, an IBM MKTTOOLS document for the z/OS UNIX System Services implementer

It is also recommended that you attend an z/OS UNIX System Services (OpenEdition) concepts class and a class in using z/OS UNIX System Services prior to migrating to IBM Communications Server for z/OS. If this is not possible, then you will want to ensure that the z/OS UNIX System Services implementer and the RACF administrator work together with you during the installation and customization process.

Planning for and installing IBM Communications Server for z/OS requires MVS, UNIX, and networking skill. If your background is in traditional MVS programming or systems programming, the z/OS UNIX System Services terminology may at first seem to be somewhat confusing. If your background is in the UNIX environment, the terms should be familiar to you.

In the past, MVS TCP/IP system programmers have needed a working knowledge of the MVS or z/OS system. These programmers have been accustomed to working closely with the RACF administrator and z/OS system programmer for authorizations, the VTAM and NCP system programmers for SNALINK and NCP connections, the IP address administrator for basic name and address assignments, and the administrators of the router network and channel-attached peripherals for connection definition and problem determination.

With the introduction of IBM Communications Server for z/OS , the TCP/IP system programmer needs to develop an additional alliance with the z/OS UNIX System Services system programmer. The TSO interfaces that have been traditionally available in the host-based TCP/IP still stand at the system programmer's disposal and new MVS console commands simplify some TCP/IP operations. However, another user interface provided by the UNIX shell environment, either with the OMVS shell or the ISPF SHELL, is a useful and sometimes necessary tool that the TCP/IP system programmer will need to work with. Additionally, the tight coupling of IBM Communications Server for z/OS with z/OS UNIX System Services means that the TCP/IP system programmer needs more than a passing knowledge of UNIX conventions, commands, and Hierarchical File System (HFS) concepts. Even if the system programmer is familiar with other UNIX environments, work with the UNIX shell requires more than basic familiarity.

In the first version of a full TCP/IP stack based on native MVS and on z/OS UNIX System Services, few have all the requisite skills to successfully implement z/OS IP on their own. As more and more systems programmers acquire skills in UNIX System Services and in TCP/IP, this will become less and less the case. Working with the z/OS UNIX System Services implementer when implementing z/OS IP provides the most effective solution to establishing a working z/OS IP environment.

Additional assistance to the z/OS UNIX System Services implementer is an z/OS Installation (OS390CKL) checklist made available to IBM employees on MKTTOOLS. This checklist is designed to aid in developing and executing complete z/OS implementation plans. It will be updated by general availability z/OS releases.

If you are migrating to IBM Communications Server for z/OS , establish a migration process to move all your existing applications, and after this, consider the use of new and enhanced functions based on *z/OS Communications Server: IP Migration*.

IBM Communications Server for z/OS allows multiple copies of the TCP/IP protocol stack to execute on the same MVS image. However, with all the performance enhancements introduced in z/OS IP, it is probably not necessary to implement a multi-stack system for production purposes unless one is considering building a system programming test stack.

You are now ready to move on to the following steps.

Step 1: Install z/OS CS

Before you begin the installation:

- Read *z/OS Planning for Installation* to help you plan the installation and migration of z/OS CS.
- Be sure you understand the data set naming conventions used in TCP/IP. You can find this information in "Configuration Data Set Naming Conventions" on page 6.
- Consult the *z/OS Program Directory* (Considerations for Wave 2A Customization) for current information about the material, procedures, and storage estimates of the MVS image.

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

Verifying the Initial Installation

Both the *z/OS Program Directory* and *z/OS Installing Your Order* contain step-by-step instructions that can be used to set up and verify a basic TCP/IP configuration with only the loopback address and a few key servers. For more information regarding these instructions, refer to the information about Wave 1A Customizations in the *z/OS Program Directory* or the information about verifying your installation in *z/OS Installing Your Order*.

Step 2: Customize z/OS CS

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

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

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

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

Making SYS1.PARMLIB Changes

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

ServerPac method

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

CBPDO method

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

z/OS CS exploits z/OS UNIX services even for traditional MVS environments and applications. Prior to utilizing TCP/IP services, therefore, a full-function mode z/OS UNIX environment—including a Data Facility Storage Management Subsystem (DFSMSdfp™), a Hierarchical File System (HFS), and a security product {such as Resource Access Control Facility (RACF)}—needs to be defined and active before z/OS CS can be started successfully.

Additional information about required TCP/IP definitions for the UNIX environment can be found in "Defining TCP/IP as a UNIX System Services Physical File System (PFS)" on page 36 and "UNIX System Services Security Considerations" on page 30.

Common z/OS UNIX Configuration Problems: Following are some explanations and possible solutions for common problems that you may encounter when configuring the z/OS UNIX environment.

- TCP/IP initialization fails with the following messages:

```
EZZ4203I OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIP-BPX1SOC,
     00000003,FFFFFFFF,00000070,112B00B6
```

  These messages usually indicate that both INET and CINET FILESYSTYPE have been specified. Only one should be specified; refer to the FILESYSTYPE section in *z/OS UNIX System Services Planning* for additional information.

- TCP/IP initialization fails with the following messages:

```
EZZ4203I OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIP-BPX1SOC,
     00000003,FFFFFFFF,0000006F,112B00B0
```

  These messages indicates that the requester of the service is not privileged. The service requested requires a privileged user. Check the documentation for the service to understand what privilege is required.

- TCP/IP initialization fails with the following messages:

```
EZZ4203I OPENEDITION-TCP/IP CONNECTION ERROR
     FOR TCPIPA-BPX1IOC,8008C981,FFFFFFFF,0000009E,12B2005A

EZZ4204I TCPIP INITIALIZATION FOR TCPIPA FAILED
```

  These messages usually indicate that an incorrect jobname was specified in the SUBFILESYSTYPE NAME() definition in the BPXPRM*xx* member for a common INET environment. In this scenario, the NAME() must match TCPIPA.

- TCP/IP initialization fails with the following messages:

```
IEA8481 DUMP SUPPRESSED - ABDUMP MAY NOT DUMP STORAG FOR KEY 0-7 JOB TCPV34A
IEF4501 TCPIPA TCPIPA - ABEND=SEC6 U0000 REASON=0F01C008
```

These messages are usually an indicator that an OMVS RACF segment has not been defined for the user ID associated with the TCP/IP started procedure. Define an OMVS segment with a UID of 0 for the user ID associated with the TCP/IP started procedure.

- TCP/IP initialization fails with the following messages:

```
IEF4031 TCPIPA - STARTED - TIME=16.01.25
EZZ42031 OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIPA-BPX11OC,
     8008139A,FFFFFFFF,00000079,12D2025E
EZZ42041 TCPIP INITIALIZATION FOR TCPIPA FAILED.

==> The 0079 value is EINVAL - The parameter is incorrect
==> The 025E value is JRSocketCallParmError - A socket syscall
    contains incorrect parameters
```

These messages usually indicate that an incorrect entry point name has been specified in the SUBFILESYSTYPE ENTRYPOINT() definition. The correct value is ENTRYPOINT(EZBPFINI).

- TCP/IP initialization fails with the following messages:

```
EZZ32031 OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIPA-BPX1SOC,
     00000003,FFFFFFFF,0000045A,112B0000
EZZ4204I TCPIP INITIALIZATION FOR TCPIPA FAILED.

==> The 045A value is EAFNOSUPPORT - The address family is not supported
```

These messages indicate that AF_INET was not defined or did not initialize properly. Check for any earlier z/OS UNIX messages and verify that the z/OS UNIX NETWORK DOMAINNAME(AF_INET) statement is in your BPXPRM*xx* member.

- After issuing a NETSTAT command from TSO, the following message is displayed:

```
netstat
CEE5101C During initialization, the z/OS UNIX callable service
     BPX1MSS failed. The system return code was 0000000156,
     the reason code was 0507014D. The application will be
     terminated.
NETSTAT ENDED DUE TO ERROR+
READY
?
USER ABEND CODE 4093 REASON CODE 00000090
READY

==> The 0156 value is EMVSINITIAL - Process initialization error
==> The 014D value is JRFsFailChdir - The dub failed, due to
    an error with the initial home directory
```

These messages indicate that the user ID issuing the NETSTAT command does not have an OMVS RACF segment defined for it. Define an OMVS segment for this user ID or activate the default OMVS segment support. For details, see “UNIX System Services Security Considerations” on page 30.

- Socket applications using the z/OS CS TCP/IP Services APIs fail with an ERRNO of 156.

  ERRNO 156 indicates an z/OS UNIX process initialization failure. This is usually an indication that a proper OMVS RACF segment is not defined for the user ID associated with the application. The RACF OMVS segment may not be defined or may contain errors such as an improper HOME() directory specification. If the OMVS segment is not defined, you may also receive the following message:

```
ICH4081 USER(USER8   ) GROUP(SYS1   )   NAME(TSO USERID USER8  )
   CL(PROCESS )
   OMVS SEGMENT NOT DEFINED
```

In this example, USER8 is the user ID associated with the failing application. To correct this problem, define a proper OMVS segment for the user ID associated with the failing application. For details, see "UNIX System Services Security Considerations" on page 30.

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

The subsequent chapters in this book show you how to:

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

  You can find a description of the MVS Component Trace support in the *z/OS Communications Server: IP Diagnosis*.
- Specify the ENVAR parameter on the PARMS=CTRACE(CTIEZB00) keyword to override the resolver file. For more information on setting the environment variable RESOLVER_CONFIG using the ENVAR parameter, see "Considerations for Multiple Instances of TCP/IP" on page 37.
- Configure each of the servers you want to run. This might require:
  - Modifying sample procedures and adding them in your PROCLIB
  - Modifying the configuration data set, PROFILE.TCPIP
  - Adding port numbers to *hlq*.ETC.SERVICES
  - Modifying other data sets containing server-specific parameters

You can find the sample procedures and data sets in *hlq*.SEZAINST or the HFS. Table 1 on page 8 provides additional reference information you can use as you configure and customize each server.

You can find general information about starting, stopping, and dynamically controlling the servers in *z/OS Communications Server: IP Configuration Reference*. Specific information about operating and administering each server is also provided in the chapter for that server.

Step 3: Configure VMCF and TNF

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

The restartable VMCF must be started before TCP/IP if you want the VMCF node name used as a default host name during TCP/IP initialization (in cases where no other host name can be located).

Note: Host name is the value normally specified on the TCPIP.DATA HOSTNAME statement.

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

Restartable Subsystems

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

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

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

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

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

   ```
   TNF
   VMCF
   ```

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

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

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

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

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

Non-Restartable Subsystems

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

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

Do not use the sample SEZAINST (IEFSSN) as shipped, because the comments are not valid in SYS1.PARMLIB. A modified form of the last two lines must be placed in the IEFSSN*xx* PARMLIB member. Replace node name on the VMCF line with the NJE node name of your MVS system.

VMCF Commands

If you will be using restartable VMCF, the following VMCF commands let you display the names of the current users of VMCF and TNF, and if necessary, remove names from the name lists.

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

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

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

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

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

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

   ```
   P VMCF
   P TNF
   ```

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

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

Following are descriptions of the commands:

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

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

P TNF Requests TNF to terminate

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

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

P VMCF
Requests VMCF to terminate

Following are sample commands:

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

Common VMCF Problems

Following are some common VMCF problems:

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

  This is probably an installation problem; check the PPT entries for errors. Some levels of MVS do not flag PPT syntax errors properly.

- Abends 0D5 and 0D6 after REMOVEing a user.

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

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

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

IUCV/VMCF Considerations

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

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

  An example of a TCP/IP-provided application that exploits AF_IUCV sockets is the SNMP Query Engine component (SQESERVE). The z/OS UNIX socket library provides a similar functionality to the AF_IUCV domain sockets with its AF_UNIX domain. Users creating new applications should consider using AF_UNIX domain sockets.
- The Pascal socket interface also makes use of the IUCV/VMCF services for a limited set of inter-address space communication flows. As a result, any applications (provided by IBM or others) that use the Pascal socket API also still have a requirement for the VMCF/TNF subsystems. TCP/IP provides several applications and commands that exploit these interfaces, such as the SMTP, LPD, and the TELNET, PING, TRACERTE, HOMETEST, TESTSITE, RSH, REXEC, and LPR commands.

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

Step 4: Update the VTAM Application Definitions

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

- SNALINK
- SNALINK LU6.2
- Telnet

- X.25 NPSI Server

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

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

Step 5: Start the TCP/IP Address Space

To verify that your configuration is correct, start the TCP/IP address space.

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

```
START TCP1
```

Step 6: Set Up Cataloged Procedures and Configuration Data Sets

At this point in the configuration process, you can choose to either set up procedures or you can do each one individually when you set up the appropriate application, function, or server.

See the remaining chapters in this book for more information about setting up the appropriate application, function, or server.

Step 7: Customize TCP/IP Messages

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

How to Access the Message Data Sets

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

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

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

Message Text

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

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

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

Message Format

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

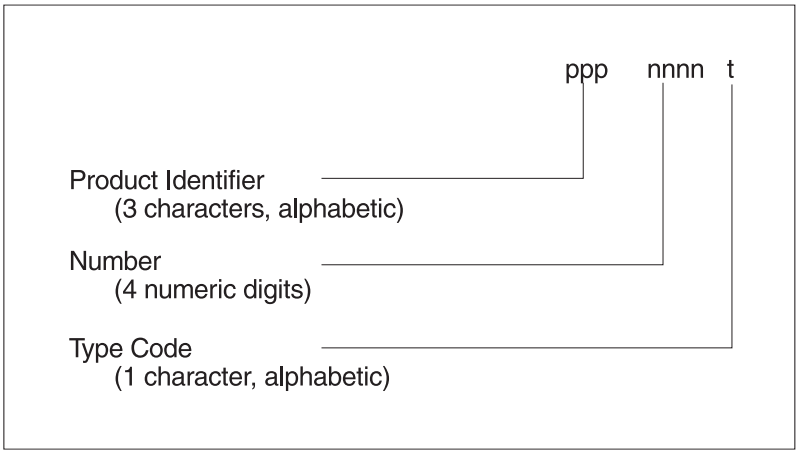


Figure 12. Syntax for TCP/IP Message IDs

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

Rules for Customizing the Messages

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 2. Customization

Before you begin customizing, it is assumed that you know what configuration data sets are used by the TCP/IP address space, their search order, and considerations for what type of TCP/IP stack you will be running in your environment (for example, Enterprise Extender (EE) and multiple stacks). See “Chapter 1. Configuration Overview” on page 3 for this information.

After reading this chapter, you will know how to configure and start syslogd and the TCP/IP stack. You should understand the relationships of TCP/IP configuration files as they apply to the TCP/IP address space. The three main configuration files that you will be working with are:

- TCPIP.DATA
- PROFILE.TCPIP
- HOSTS.LOCAL

You should be able to use the following commands to verify customization:

PING Sends IP datagrams to a specified destination host, requesting a reply, and measures the round trip time. This helps you to verify the interfaces defined to the TCP/IP address space.

NETSTAT
Queries TCP/IP about the network status of the local host. With NETSTAT, you can verify **most** TCP/IP customization values that can be set from the PROFILE.TCPIP.

HOMETEST
Verifies your host name and address configuration. This is a TSO command.

TRACERTE
Displays the route that a packet takes to reach a requested destination.

Configuring the Syslog Daemon (syslogd)

Configuration Statements

The syslogd processing is controlled by a configuration file called /etc/syslog.conf (see the following sample file) in which you define logging rules and output destinations for error messages, authorization violation messages, and trace data. Logging rules are defined using a *facility* name, a *priority* code, and the user ID and job name of the program that generated the message. The *facility* name and *priority* code are passed on the logging request from an application when it wants to log a message. The user ID and job name are provided by the system. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about logging rules.

As shown in the following sample /etc/syslog.conf file, comments can be added to the configuration file by placing the # character in column one of the comment line. Everything following the hashmark (#) character is treated as a comment.

```
#  Licensed Materials - Property of IBM
#  "Restricted Materials of IBM"
#  5647-A01
#  (C) Copyright IBM Corp. 1995, 2000
#  Status = CSV2R10
#
```

```
# /etc/syslog.conf - control output of syslogd
#
# The # sign begins a comment which extends to the end of the line.
#
# Blank lines are ignored.
#
# Rules in this file specify types of messages which syslogd will
# store, and where syslogd will store it.
#
# See IP Configuration Reference for detailed information about
# the syntax.  These comments are meant to provide only a general
# overview.
#
# Four criteria can be used to select messages for processing:
#
# 1) user ID associated with application generating the message
#
#    * can be specified for the user ID if the user ID is not
#    important.
#
# 2) job name of application generating the message
#
#    * can be specified for the job name if the job name is not
#    important.
#
# 3) facility of the message, as specified by the application
#
#    This is user, mail, news, uucp, daemon, auth, cron, lpr, or
#    local0-local7.  Consult the documentation for the application
#    to determine which facility the application specifies.
#
#    A special facility, mark, specifies that syslogd should log
#    mark messages on a regular basis.  These can be used to verify
#    that syslogd was operational during a specific time interval.
#
# 4) priority of the message, as specified by the application
#
#    This is emerg, panic, alert, crit, err, error, warn, warning,
#    notice, info, or debug.
#
#    A special priority, none, specifies that messages with the
#    specified user ID, job name, or facility should not be
#    selected.
#
# These criteria are specified together as
#
#    userid.jobname.facility.priority
#
# or, if user ID and job name are both *, as
#
#    facility.priority
#
# This can be combined in a series as
#
#   userid.jobname.facility.priority;userid.jobname.facility.priority
#
# The criteria for selecting messages for processing are combined
# with a destination, which tells syslogd what to do with selected
# messages.
#                       #   criteria    destination
#
# The destination can be a file, one or more user IDs, SMF, syslogd
# at a remote host, or all logged-in users.
#
# The following example stores messages with facility daemon or
# local1 in the file /directory/logfile.
#
```

```
#   daemon.*;local1.*   /directory/logfile
#
# The directory structure used in this sample configuration is
# expected to be created automatically by syslogd, with a new
# directory of log files for each day.  This requires two types
# of configurations outside of the scope of this configuration
# file:
#
# 1) syslogd command-line option
#
#    The syslogd -c command-line option should be enabled, causing
#    syslogd to create log files and directories if they do not
#    already exist.
#
# 2) cron job
#
#    A cron job should be utilized to wake up syslogd at the
#    beginning of each day to switch to new log files in a new
#    directory.  Here is the cron job definition:
#
#      1 0 * * * kill -HUP 'cat /etc/syslog.pid'
#
#    This job should be defined for a user ID with UID zero so that
#    it has permissions to send the signal to syslogd.
#
#    See UNIX System Services Planning and UNIX System Services
#    Command Reference for more information about cron.
#
# A sample shell script is provided for removing log files which are
# a specified number of days old.  It assumes the same directory
# structure which is used in this sample configuration.
#
# All example rules except for the last one are commented-out.  Some
# or all of the example rules will need to be changed for your
# environment.  Each example rule contains an explanation of changes
# which may be required.
#
#####################################################################
#
# Write all messages with priority crit or higher to the MVS operator
# console.  See the UNIX System Services Planning manual for more
# information about the /dev/console special file.
#
# *.crit             /dev/console
#
```

Starting and Stopping syslogd

Following is the syntax for the syslogd command:

syslogd [–f *conffile***] [–i][–u[–c[–d][–m** *markinterval***] [–p** *logpath***]**

syslogd recognizes the following options:

-c Create log files and directories automatically.

-d Run syslogd in debugging mode (see “Diagnosing syslogd Configuration Problems” on page 68 for more information).

-f Configuration file name.

-i Do not receive messages from the IP network.

-m Number of minutes between mark messages. The default value is 20 minutes.

-p Path name of z/OS UNIX character device for the datagram socket. The default value is /dev/log.

Note: This option is not used frequently. If you selected the -p option, syslogd will not function properly.

-u For records received over the AF_UNIX socket (most messages generated on the local system), include the user ID and job name in the record. In this case, a forward slash, the user ID, and the job name will follow the local host name for messages received over the AF_UNIX socket. The forward slash, which immediately follows the local host name, can be used to determine whether or not the user ID and job name is being recorded. If not recorded, a blank immediately follows the local host name. When user ID or job name is not available, *N/A* will be written in the corresponding field.

To specify the job name and pass the appropriate environment variables to the syslogd process, start syslogd using a shell script such as the following:

```
#
# Start the syslog daemon
#

export _BPX_JOBNAME='syslogd'
/usr/sbin/syslogd -f /etc/syslog.conf &
```

You can execute this shell script directly from the /etc/rc file to start syslogd at z/OS UNIX initialization.

If an invalid argument or number of arguments is entered, syslogd exits and the return code is 1. In all other situations in which syslogd exits, the return code is 0 (zero).

To terminate syslogd, send a SIGTERM signal.

```
kill -s SIGTERM <PID>
```

To force syslogd to reread its configuration file and activate any modified parameters without stopping, send a SIGHUP signal. syslogd will continue to append log messages to the files you specify in /etc/syslog.conf.

```
kill -s SIGHUP <PID>
```

The syslog daemon stores its process ID in the /etc/syslog.pid file so that it may be used to terminate or reconfigure the daemon. For syslogd to successfully create this file, you must define the syslogd user ID as UID=0.

Note: If the BPX.SMF facility is defined and SMF records are to be written by syslogd, the user ID with which syslogd runs must also be permitted to SAF resource BPX.SMF. See SEZAINST(EZARACF) for more information.

Messages are read from the UNIX domain datagram socket and, unless the -i command line option is specified, the Internet domain datagram socket. Kernel messages are not logged by syslogd in z/OS UNIX.

Note: For more information about the facilities used by z/OS CS functions, see Table 3 on page 25.

Offloading Log Files

z/OS CS includes a syslogd configuration file in /usr/lpp/tcpip/samples/syslog.conf, a REXX program for removing old log files in /usr/lpp/tcpip/samples/rmoldlogs, and a JCL procedure for starting syslogd in SEZAINST(syslogd). These are intended to be used together, though each may need to be customized for your installation.

The sample syslogd configuration file is installed in /usr/lpp/tcpip/samples/syslog.conf. It can be copied to /etc/syslog.conf after customization. If it is copied somewhere else, the syslogd -f command-line option must be used to tell syslogd where to find the configuration file.

The sample REXX program for removing old log files is installed in /usr/lpp/tcpip/samples/ezaslrol. It can be copied to an installation-defined directory after customization. The sample JCL procedure can be copied to an installation-defined library after customization.

The sample configuration uses date stamps in the names of directories of log files to organize log files by year, month, and day. Log files for February 14, 2001 would be stored in directory /var/log/2001/02/14.

A cron job should be used to send the SIGHUP signal to syslogd every day at midnight so that it switches to a new set of files. The cron job should be created for a user ID with UID 0. The definition of the cron job is:

```
0 0 * * * kill -HUP 'cat /etc/syslog.pid'
```

The log file names vary based on the day, so sending SIGHUP to syslogd after the day changes causes syslogd to create new files.

Because some messages sent just after midnight may be logged by syslogd before it processes the SIGHUP signal, it is possible that a few messages sent after midnight will be stored in the log files for the previous day.

The sample REXX program can be run daily to remove all log files older than the number of days specified in the program. Comments in the REXX program describe how to configure the number of days. The definition of a cron job to run the REXX program every day at 1:00 A.M. is:

```
0 1 * * * localdir/ezaslrol
```

localdir is the name of the installation-defined directory where the customized version of /usr/lpp/tcpip/sample/ezaslrol was copied.

Using syslogd for z/OS UNIX Application Programs

You can use the logging facilities of the syslogd server with your z/OS UNIX application programs. Include the syslog.h header file with C programs so that they can open a log facility, send log messages to syslogd, and close the facility:

```
#include <syslog.h>

1 openlog("oec", LOG_PID, LOG_LOCAL0);
2 syslog(LOG_INFO, "Hello from oec");
3 closelog();
```

1 Open a log facility with the name of local0. Prefix each line in the log file with the program name (oec) and the process ID.

2 Log an info priority message with the specified content.

3 Close the log facility name.

The preceding statements created the following line in the log file:

```
May 26 11:27:51 mvs18oe oec[3014660]: Hello from oec
```

For more information about the syslog function, refer to *Advanced Programming in the UNIX Environment*, published by Addison-Wesley or *z/OS C/C++ Run-Time Library Reference*.

Usage Notes

- syslogd can be started only by a task or user with superuser authority.
- syslogd can be terminated using the SIGTERM signal.
- If you want syslogd to receive log data from remote syslogd servers, ensure that syslogd can bind to UDP port 514 by reserving that port for the syslogd job in your PROFILE.TCPIP data set. Ensure that the syslog service is defined in your services file or data set (for example, /etc/services). The following example port reservation in PROFILE.TCPIP assumes that syslogd runs as job syslogd1:

  ```
  PORT
  ...
  514 UDP syslogd1     ;syslogd daemon
  ...
  ```

  The following example shows the services file or data set file entry:

  ```
  syslog     514/udp
  ```

- If there is no TCP/IP transport active when syslogd starts or if TCP/IP is recycled, syslogd will establish or reestablish communication with TCP/IP when it becomes available.
- Configuration file errors are written to the operator console because initialization is not complete until the entire configuration file has been read.
- Facility mark is not affected by the *.priority usage. Mark messages are written only to the destinations of rules that specify mark.info.
- If a mark interval of zero minutes is specified, mark messages will be written every thirty seconds.

Diagnosing syslogd Configuration Problems

syslogd supports a debug mode, which is selected using the -d command-line option. In this debug mode, syslogd does not run as a daemon, but instead runs in the foreground and writes a large number of trace messages to STDOUT. These messages can be used to diagnose problems in the syslogd configuration or to collect documentation when reporting a syslogd problem to IBM support.

Note: Do not use the -d option for normal operations.

If you are running syslogd in batch with -d, debug output is written to SYSPRINT, SYSTERM, or SYSERR, whichever is found first. The sample syslogd procedure SEZAINST(syslogd) defines SYSPRINT so that debug messages are stored in the job output.

Use caution using -d when syslogd is started from /etc/rc. If -d is used in this way, the shell *and* operator must be used to run syslogd in the background. Otherwise, /etc/rc does not end and UNIX System Services initialization does not complete.

Also, use caution when using -d along with a port reservation statement for the syslogd port (UDP port 514) in the TCP/IP profile. The job name of syslogd might differ based on whether or not the -d option was specified. If a port reservation statement is coded based on the job name that syslogd uses without the -d option, syslogd might not be able to bind to the port when run with -d. When using debug mode with a port reservation statement for the wrong job name, the bind() error can be ignored or the -i command-line option can be specified along with -d so that syslogd will not get a UDP socket.

Configuring TCPIP.DATA

Use of TCPIP.DATA and /etc/resolv.conf

The TCPIP.DATA configuration data set is the anchor configuration data set for the TCP/IP stack and all TCP/IP servers and clients running in z/OS. In z/OS IP, you can define the TCPIP.DATA parameters in an HFS file or in an MVS data set. The TCPIP.DATA configuration data set is read during initialization of all TCP/IP server and client functions. All functions must access this data set in order to find basic configuration information, such as the name of the TCP/IP address space, the TCP/IP host name, and the data set prefix to use when searching for other configuration data sets.

The TCPIP.DATA data set is also known as one of the resolver configuration data sets. In fact, this name is now more commonly used to refer to this important file in the UNIX System Services environment because the socket library contains a component called the resolver. In a UNIX system, you use the /etc/resolv.conf file for the same purpose as you use TCPIP.DATA in your MVS system.

These files specify the name of the TCP/IP address space. Because the data set search order can vary, your installation will determine which data set you can use. See “Chapter 1. Configuration Overview” on page 3 for search order, data set, and file retrieval information and “Resolver Configuration Files” on page 15.

If you use TCPIP.DATA, it can be shared between multiple systems with a system name. But, if TCPIP.DATA is allocated via SYSTCPD DD and an application forks, any allocations from the parent of SYSTCPD are lost to the child process.

If you use /etc/resolv.conf instead of TCPIP.DATA, each application can have its own environment variable, RESOLVER_CONFIG=’xxx’. There are no concerns for forked child processes; however, this means that you cannot share the same data set or file among multiple systems.

Creating TCPIP.DATA

Create a TCPIP.DATA file by copying the sample provided in SEZAINST(TCPDATA) and modifying it to suit your local conditions.

Allocate this data set with either sequential (PS) or partitioned (PO) organization, a fixed block format (FB), a logical record length (LRECL) of 80, and any valid block size value for a fixed block, such as 3120. This file can also be the HFS file /etc/resolv.conf, or an HFS file that is pointed to by either the environment variable RESOLVER_CONFIG or the SYSTCPD DD in a JCL procedure. The environment variable RESOLVER_CONFIG can also point to an MVS data set or PDS.

Note: The EZASMI interface expects the LRECL for a Fixed Block file to be less than 256.

You can use any name for the TCPIP.DATA data set if you access it using the //SYSTCPD DD statement, or use ENVAR to set RESOLVER_CONFIG, in the JCL for all the servers, logon procedures, and batch jobs that execute TCP/IP functions. If you are not using the //SYSTCPD DD statement, the environment variable, or /etc/resolv.conf, then the data set name must conform to the conventions described in “Configuration Files for the TCP/IP Stack” on page 12. Another alternative is to use the well-known data set name SYS1.TCPPARMS(TCPDATA). You will issue the HOMETEST command to verify the actual data set name the system finds for TCPIP.DATA later in this chapter. However, because HOMETEST is an MVS sockets application, it does not use RESOLVER_CONFIG or /etc/resolv.conf in its search order. For this reason, it is recommended that /etc/resolv.conf and TCPIP.DATA contain exactly the same information.

TCPIP.DATA Statements

Each configuration statement can be preceded by an optional *system_name*. This permits configuration information for multiple systems to be specified in a single *hlq*.TCPIP.DATA data set. The *system_name* is matched against the name of the system on which you are running. The name of the system is taken from the node name in the IEFSSNxx member of parmlib, which is the third parameter of the VMCF line.

The statements are processed in the order they appear in the data set. The following rules apply to this processing:

- If the *system_name* does not match the name of the system, the configuration statement is ignored.
- If *system_name* is blank, the configuration statement is in effect on every system.
- If the *system_name* matches the node name, the configuration statement that follows it is in effect.
- The ***last*** statement that matches is effective.

For example, if you have the following three TCPIPJOBNAME statements, MVS6 would look for a TCP/IP cataloged procedure named TCPBTA2, MVSA would look for TCPV3, and all other systems would look for TCPMCWN.

```
      TCPIPJOBNAME TCPMCWN
MVS6: TCPIPJOBNAME TCPBTA2
MVSA: TCPIPJOBNAME TCPV3
```

But if you reversed the order, all systems would try to find the procedure named TCPMCWN.

```
MVS6: TCPIPJOBNAME TCPBTA2
MVSA: TCPIPJOBNAME TCPV3
      TCPIPJOBNAME TCPMCWN
```

Sample TCPIP.DATA Data Set (TCPDATA)

The following sample is used to specify configuration information of client parameters.

```
;*********************************************************************
;                                                                    *
;   Name of Data Set:     TCPIP.DATA                                 *
;                                                                    *
;   COPYRIGHT = NONE.                                                *
;                                                                    *
;   This data, TCPIP.DATA, is used to specify configuration          *
;   information required by TCP/IP client and server programs.       *
;                                                                    *
;                                                                    *
;   Syntax Rules for the TCPIP.DATA configuration data set:          *
```

```
;                                                                      *
;   (a) All characters to the right of and including a ; will be       *
;       treated as a comment.                                          *
;                                                                      *
;   (b) Blanks and  are used to delimit tokens.                        *
;                                                                      *
;   (c) The format for each configuration statement is:                *
;                                                                      *
;         keyword  value                                               *
;                                                                      *
;       where  is an optional label that can be                        *
;       specified before a keyword; if present, then the keyword-      *
;       value pair will only be recognized if the SystemName matches   *
;       the node name of the system, as defined in the IEFSSNxx        *
;       PARMLIB member.  This optional label permits configuration     *
;       information for multiple systems to be specified in a single   *
;       TCPIP.DATA data set.                                           *
;                                                                      *
;       NOTE:  You should define the SystemName in the IEFSSNxx        *
;              PARMLIB member to be the same as your JES2 or JES3      *
;              node name.  This is required for correct delivery of    *
;              SMTP mail.                                              *
;                                                                      *
;   (d) If a statement is specified multiple times, the last statement *
;       is effective.                                                  *
;                                                                      *
;***********************************************************************
;
; TCPIPJOBNAME statement
; ======================
; TCPIPJOBNAME specifies the name of the started procedure that was
; used to start the TCPIP address space.    TCPIP is the default.
;
; If multiple TCPIP stacks are run on a single system, each stack will
; require its own copy of this file, each with a different value for
; TCPIPJOBNAME.
;
TCPIPJOBNAME TCPIP
;
;
; HOSTNAME statement
; ==================
; HOSTNAME specifies the TCP host name of this system as it is known
; in the IP network.  If not specified, the default HOSTNAME will be
; the node name specified in the IEFSSNxx PARMLIB member.
;
; For example, if this TCPIP.DATA data set is shared between 2
; systems, OURMVSNAME and YOURMVSNAME, then the following 2 lines
; will define the HOSTNAME correctly on each system.
;
; OURMVSNAME:    HOSTNAME  OURTCPNAME
; YOURMVSNAME:   HOSTNAME  YOURTCPNAME
;
; No prefix is required if the TCPIP.DATA file is not being shared.
;
HOSTNAME THISTCPNAME
;
; DOMAINORIGIN statement
; ======================
; DOMAINORIGIN specifies the domain origin that will be appended
; to host names passed to the resolver.  If a host name contains
; any dots, then the DOMAINORIGIN will not be appended to the
; host name.
;
DOMAINORIGIN  YOUR.DOMAIN.NAME
;
;
```

```
; DATASETPREFIX statement
; ======================
; DATASETPREFIX is used to set the high level qualifier for dynamic
; allocation of datasets in TCP/IP.
;
; The character string specified as a parameter on
; DATASETPREFIX takes precedence over the default prefix of "TCPIP".
;
; The DATASETPREFIX parameter can be up to 26 characters long
; and the parameter must NOT end with a period.
;
; For more information please see "Dynamic Data Set Allocation" in
; the IP Configuration Guide.
;
DATASETPREFIX TCPIP
;
;
; MESSAGECASE statement
; =====================
; MESSAGECASE MIXED indicates to some servers, such as FTPD, that
; messages should be displayed in mixed case.  MESSAGECASE UPPER
; indicates that all messages should be displayed in uppercase.  Mixed
; case strings that are inserted in messages will not be uppercased.
;
; If MESSAGECASE is not specified, mixed case messages will be used.
;
; MESSAGECASE MIXED
; MESSAGECASE UPPER
;
;
; NSINTERADDR statement
; =====================
; NSINTERADDR specifies the IP address of the name server.
; LOOPBACK (127.0.0.1) specifies your local name server.  If a name
; server will not be used, then do not code an NSINTERADDR statement.
; (Comment out the NSINTERADDR line below).  This will cause all names
; to be resolved via site table lookup.
;
; The NSINTERADDR statement can be repeated as many times as you need
; to specify alternate name servers.  The name server listed first
; will be the first one attempted.
;
; NSINTERADDR  127.0.0.1
;
;
; NSPORTADDR statement
; ====================
; NSPORTADDR specifies the foreign port of the name server.
; 53 is the default value.
;
; NSPORTADDR 53
;
;
; RESOLVEVIA statement
; ====================
;
; RESOLVEVIA specifies how the resolver is to communicate with the
; name server.  TCP indicates use of TCP connections.  UDP indicates
; use of UDP datagrams.  The default is UDP.
;
RESOLVEVIA UDP
;
;
; RESOLVERTIMEOUT statement
; =========================
; RESOLVERTIMEOUT specifies the time in seconds that the resolver
; will wait for a response from the name server (either UDP or TCP).
```

```
; The default is 30 seconds.
;
RESOLVERTIMEOUT 10
;
;
; RESOLVERUDPRETRIES statement
; ============================
;
; RESOLVERUDPRETRIES specifies the number of times the resolver
; should try to connect to the name server when using UDP datagrams.
; The default is 1.
;
RESOLVERUDPRETRIES 1
;
;
; LOADDBCSTABLES statement
; ========================
; LOADDBCSTABLES indicates to the FTP server and FTP client which DBCS
; translation tables should be loaded at initialization time. Remove
; from the list any tables that are not required. If LOADDBCSTABLES is
; not specified, no DBCS tables will be loaded.
;
; LOADDBCSTABLES JIS78KJ JIS83KJ SJISKANJI EUCKANJI HANGEUL KSC5601
; LOADDBCSTABLES TCHINESE BIG5 SCHINESE
;
;
; SOCKDEBUG statement
; ===================
; SOCKDEBUG will cause a trace of socket library calls to be written.
; This command is for debugging purposes only.
;
; SOCKDEBUG
;
;
; SOCKNOTESTSTOR statement
; ========================
; SOCKTESTSTOR is used to check socket calls for storage access errors
; on the parameters to the call.  SOCKNOTESTSTOR stops this checking
; and is better for response time.  SOCKNOTESTSTOR is the default.
;
; SOCKTESTSTOR
; SOCKNOTESTSTOR
;
;
; TRACE RESOLVER statement
; ========================
; TRACE RESOLVER will cause a complete trace of all queries to and
; responses from the name server or site tables to be written to
; the user's console.  This command is for debugging purposes only.
;
; TRACE RESOLVER
;
;
; TRACE SOCKET statement
; ======================
; TRACE SOCKET will cause a complete trace of all calls to TCP/IP
; though the C socket library to the SYSPRINT DD dataset.
; This statement is for debugging purposes only.
;
; TRACE SOCKET
;
;
; ALWAYSWTO statement
; ===================
; ALWAYSWTO causes messages for some servers, such as SMTP and LPD,
; to be issued as WTOs.
;
```

```
; ALWAYSWTO YES
;
; Obsolete statements
; ==================
; The following statements no longer have any effect when included in
; this file:
;   SOCKBULKMODE
;   SOCKDEBUGBULKPERF0
;
; End of file.
;
```

TCPIPJOBNAME

The TCPIPJOBNAME is the job name of the TCP/IP address space with which the application or end user wants to communicate. In this example, the TCP/IP address space name is TCPIP. If the TCPIPJOBNAME in the TCPIP.DATA data set or /etc/resolv.conf allocated to an application or end user is incorrectly specified, many applications cannot communicate with TCP/IP. In the case of NETSTAT, if the TCPIPJOBNAME is incorrectly set to TCPXYZ, the end user sees the message `EZZ2382I Unable to open UDP socket to TCPXYZ: TCPXYZ is not active`.

HOSTNAME

HOSTNAME specifies the host portion of the DNS name of the system. If a name is not specified, the node name specified in the IEFSSNxx PARMLIB member is used.

DOMAINORIGIN

The statement is the default domain name used for DNS searches. For example, if the complete host name is TCPIP.MYCOMPANY.COM, then the DOMAINORIGIN is MYCOMPANY.COM.

DATASETPREFIX

DATASETPREFIX is the high level qualifier (HLQ) used when dynamically allocating data sets.

MESSAGECASE

MESSAGECASE is used to indicate whether messages should appear in MIXED or UPPER case for servers that support the MESSAGECASE statement.

NSINTERADDR

NSINTERADDR is used to specify the IP address of the DNS server. Multiple NSINTERADDR statements can be used to specify a search order if the first name server is not found. To signify a name sever on the local system, use 127.0.0.1 (the LOOPBACK address). Additionally, for name server configuration, NSPORTADDR can be used to specify what port the name server is listening on (53).

RESOLVEVIA

The RESOLVEVIA statement can be used to specify whether the resolver should communicate with the name server using TCP or UDP.

RESOLVERTIMEOUT

The RESOLVERTIMEOUT statement can be used to specify the number of seconds the resolver waits for a response from the name server.

RESOLVERUDPRETRIES

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

TRACE RESOLVER
TRACE RESOLVER can be used for debugging purposes to produce a complete trace of all queries to and responses from the name server or site table. For performance reasons, TRACE RESOLVER should not be used except during debugging.

LOADDBCSTABLES
The LOADDBCSTABLES statement indicates to the FTP server and client which DBCS translation tables should be loaded at initialization time. No DBCS tables loaded is the default. Thus, if DBCS translation is needed, list all the tables that are required for the FTP server and client.

SOCKDEBUG
For socket level debugging, the SOCKDEBUG statement can be specified to trace socket library calls.

SOCKTESTSTOR
The SOCKTESTSTOR statement can be used to check socket calls for storage access errors on the parameters to the socket call.

TRACE SOCKET
The TRACE SOCKET statement causes a complete trace of all calls to the TCPIP address space through the C socket library. For performance reasons, TRACE SOCKET should not be used except during debugging.

ALWAYSWTO
The ALWAYSWTO can be used for some servers (like SMTP and LPD) to issue their messages via WTOs.

Creating /etc/resolv.conf

The z/OS UNIX resolver looks for /etc/resolv.conf in its search for the resolver configuration file. See “Resolver Configuration Files” on page 15 for more information.

Because different applications can use different resolvers to locate key information, it is recommended that the information in TCPIP.DATA and /etc/resolv.conf remain identical. Certain statements in TCPIP.DATA are supported by some resolver libraries, but not by others. The search statement is supported by the onslookup resolver, but it is not supported by the native MVS resolver or the z/OS UNIX resolver. The use of search can lead to inconsistent results among applications that use different resolvers.

The NAMESERVER and DOMAIN keywords are supported by the onslookup resolver and the z/OS UNIX resolver, but they are not supported by the native MVS resolver. Therefore, it is preferable to use the NSINTERADDR and DOMAINORIGIN keywords, because they are supported by all resolvers. Using NAMESERVER and DOMAIN can lead to inconsistent results among applications that use different resolvers.

Configuring PROFILE.TCPIP

During TCP/IP address space initialization, a configuration profile data set (PROFILE.TCPIP) reads system operation and configuration parameters. A sample data set, SEZAINST(SAMPPROF), can be copied and modified for use as your default configuration profile.

If you are not familiar with the search order for this data set, see “PROFILE.TCPIP Search Order” on page 12 for information about understanding data set search

orders. Refer to *z/OS Communications Server: IP Configuration Reference* for the complete statement syntax and descriptions of the configuration statements.

For ease of management when configuring a complex environment, you can use one of the following PROFILE.TCPIP data set features:

- Group related statements into separate files and use the INCLUDE statement in PROFILE.TCPIP to include them in your configuration.
- Use MVS system symbols (such as &SYSCLONE, &SYSNAME, and &SYSPLEX). Because TCP/IP translates these symbols as it reads this file, this feature reduces the number of PROFILE.TCPIP data sets that must be maintained in a multi-TCP/IP environment.

Note: For detailed information about symbols and how to define them, refer to *z/OS MVS Initialization and Tuning Reference*.

The PROFILE data set contains the following major groups of configuration parameters:

- TCP/IP operating characteristics
- TCP/IP physical characteristics
- TCP/IP reserved port number definitions (application configuration)
- TCP/IP network routing definitions
- TCP/IP diagnostic data statements

This chapter discusses the first three areas of configuration. For routing configuration information, see “Chapter 4. Routing” on page 143. For information about configuring diagnostic statements, see ADD CROSS-REFERENCE HERE.

Changing Configuration Information

If you want to change the TCP/IP configuration without stopping and starting the TCP/IP address space, you can dynamically change many of the TCP/IP configuration options established by the PROFILE.TCPIP data set. To do this, put the changed configuration statements in a separate data set and process it with the VARY TCPIP,,OBEYFILE command.

For more information about VARY TCPIP, refer to *z/OS Communications Server: IP Configuration Reference*. Also, see the Modifying section in each configuration statement in *z/OS Communications Server: IP Configuration Reference* for a description of how to dynamically change the information for that configuration statement.

Setting Up TCP/IP Operating Characteristics in PROFILE.TCPIP

The following example shows how to set up TCP/IP operating characteristics in PROFILE.TCPIP. For detailed information about any of the statements, refer to *z/OS Communications Server: IP Configuration Reference*.

Figure 13. Example of TCP/IP Operating Characteristics in PROFILE.TCPIP

```
; ====================================================================
; General TCP/IP address space configuration
; ====================================================================
;
; ARPAGE: Specifies the number of minutes between creation or
;   revalidation of an LCS ARP table entry and the deletion of the
;   entry.
```

```
;
ARPAGE 20
;
;
; ASSORTEDPARMS: Passes initialization parameters to TCPIP.  However,
;   use of this statement is being phased out.  Use the GLOBALCONFIG,
;   IPCONFIG, TCPCONFIG, and UDPCONFIG statements instead.
;
; GLOBALCONFIG: Provides settings for the entire TCP/IP stack
;
GLOBALCONFIG NOTCPIPSTATISTICS
;
; IPCONFIG: Provides settings for the IP layer of TCP/IP.
;
; Example IPCONFIG for single stack/single system:
;
IPCONFIG DATAGRAMFWD VARSUBNETTING SYSPLEXROUTING
;
; Example IPCONFIG for automatic activation of inter-stack dynamic XCF
;   and Same Host (IUTSAMEH) links
;
; IPCONFIG DYNAMICXCF 201.1.10.10 255.255.255.0 2
;
; KEEPALIVEOPTIONS: Specifies operating parameters of the TCP keep-alive
;   packets.  These same parameters can be specified on the TCPCONFIG
;   statement as well; use of the TCPCONFIG statement is recommended.
;
;
; SOMAXCONN: Specifies maximum length for the connection request queue
;   created by the socket call listen().
;
SOMAXCONN 10
;
;
; TCPCONFIG: Provides settings for the TCP layer of TCP/IP.
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to APF authorized or superuser applications.
;
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
TCPCONFIG RESTRICTLOWPORTS
;
;
; UDPCONFIG: Provides settings for the UDP layer of TCP/IP
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to APF authorized or superuser applications.
;
UDPCONFIG RESTRICTLOWPORTS
;
;
```

The following section explains the grouping of statements shown in Figure 13 on page 76.

ARPAGE

Use ARPAGE to set the number of minutes between a revalidation and deletion of ARP table entries for LCS devices. An installation that wants to describe this value in seconds versus minutes should use the IPCONFIG ARPTO statement.

Note: The ATM ARP requests are controlled via the ATMLIS statement, and the MPCIPA and MPCOSA ARP requests are not controlled by the TCP/IP address space.

ASSORTEDPARMS

It is recommended that you use the IPCONFIG, UDPCONFIG,

GLOBALCONFIG and TCPCONFIG statements rather than the ASSORTEDPARMS statement due to the side effects of the ASSORTEDPARMS statement. If some, but not all, of the ASSORTEDPARMS are specified, by default those not specified are set to OFF. Use of IPCONFIG, UDPCONFIG, GLOBALCONFIG and TCPCONFIG eliminate this side effect.

GLOBALCONFIG

Use GLOBALCONFIG to print out several counters in text format. These counters include number of TCP retransmissions and total number of TCP segments sent from the TCPIP system. Most installations will use the SMF facility of MVS to collect these counters in a more standard way.

IPCONFIG

Use IPCONFIG to configure various settings of the IP layer of TCP/IP. Use ARPTO to specify the ARP time out value in seconds for LCS devices. See 77 for more information.

Use CLAWUSEDOUBLENOP on vendor devices that document the need for double NOPs on each CCW.

Use DATAGRAMFWD if this TCP/IP is to be a router and needs to forward datagrams to other routers. Use IGNOREREDIRECT when a dynamic routing program is used and ICMP redirect packets are to be ignored by the TCP/IP address space. MULTIPATH is used to inform TCP/IP how to distribute traffic across equal cost routes. VARSUBNETTING allows the TCP/IP routing table to have address routes within the same subnet to have differing subnet masks.

Use FIREWALL to restrict this host to be a network firewall.

Applications within the network use the source IP address of a datagram to determine the source application of the datagram. When SOURCEVIPA is set, outbound datagrams use the corresponding virtual IP address (VIPA) in the HOME list instead of the physical interfaces IP address. SOURCEVIPA has no effect on RIP servers such as OROUTED, NCPROUTE, or OMPROUTE.

Use SYSPLEXRouting to communicate interface changes within a sysplex domain to the workload manager (WLM). DYNAMICXCF allows the cross communication facility within a sysplex to dynamically generate connections within a sysplex domain. If DYNAMICXCF is used with a routing program like OMPROUTE or OROUTED, then the BSDROUTINGPARMS and the OMPROUTE configuration file needs to be updated with subnet mask and cost information.

Use REASSEMBLYTIMEOUT to specify the TCP/IP reassemble timeout value in seconds and the TTL specifies the TCP/IP time to live or hop count value.

Use PATHMTUDISCOVERY to indicate to TCP/IP that it is to dynamically discover the path MTU, which is the minimum of MTUs of each hop in the path.

Use STOPONCLAWERROR to indicate to the TCP/IP stack to stop channel programs (HALTIO and HALTSIO) when a device error is detected.

KEEPALIVEOPTIONS

Use KEEPALIVEOPTIONS to specify operating parameters of the TCP

keep-alive packets. These same parameters can be specified on the TCPCONFIG statement as well; use of the TCPCONFIG statement is recommended.

SOMAXCONN
Use SOMAXCON to specify the maximum number of sockets queued on a listener.

TCPCONFIG
Use TCPCONFIG to configure various settings of the TCP protocol layer. If a keep-alive value other than two minutes is needed by an installation, use the INTERVAL statement to change the default keep-alive value. SENDGARBAGE will cause the keep-alive packet to contain one byte of random data and an incorrect sequence number, assuring that the data is not accepted by the remote TCP.

If RESTRICTLOWPORTS is specified, only applications that meet at least one of the following criteria are allowed to bind to low ports (1–1023):

- The port is reserved for the application via the PORT or PORTRANGE statement.
- The application runs with APF authorization.
- The application runs with effective POSIX UID zero.

If an installation wants to control TCP buffering (to limit storage usage or to manage large bandwidth devices), use the TCPSENDBFRSIZE, TCPRCVBUFRSIZE, and TCPMAXRCVBUFRSIZE parameters.

UDPCONFIG
Use UDPCONFIG to configure various settings of the UDP protocol layer. NOUDPCHKSUM can be used to eliminate check summing overhead for UDP packets.

If RESTRICTLOWPORTS is specified, only applications that meet at least one of the following criteria are allowed to bind to low ports (1–1023):

- The port is reserved for the application via the PORT or PORTRANGE statement.
- The application runs with APF authorization.
- The application runs with effective POSIX UID zero.

If an installation wants to control UDP buffering (to limit storage usage or to manage large bandwidth devices), use the UDPSENDBFRSIZE and UDPRCVBUFRSIZE parameters. UDPQUEUELIMIT can be used to set a queue limit for UDP. This is useful for installations that want to limit the size of the queue of UDP datagrams that an application can have waiting before the TCP/IP address space starts discarding them.

Setting Up Physical Characteristics in PROFILE.TCPIP

The following example shows how to set up physical characteristics in PROFILE.TCPIP For more information about any of these statements, refer to *z/OS Communications Server: IP Configuration Reference*.

Figure 14. Example of Physical Characteristics in PROFILE.TCPIP

```
;
; PROFILE.TCPIP
; =============
;
```

```
; This is a sample configuration file for the TCPIP address space
;
; SMP/E name: EZAEB025, alias SAMPPROF in target library SEZAINST
;
; COPYRIGHT = NONE
;
; Notes:
;
; - The device configuration, home and routing statements MUST be
;   changed to match your hardware and software configuration.
;   Likewise, the BEGINVTAM section MUST be changed to match your
;   VTAM configuration.
;
; - Lines beginning with semi-colons are comments.  To use a line
;   for your configuration, remove the semi-colon.
;
; - For more information about this file, see the IP Configuration Guide
;
; ====================================================================
; General TCP/IP address space configuration
; ====================================================================
;
; ARPAGE: Specifies the number of minutes between creation or
;   revalidation of an LCS ARP table entry and the deletion of the
;   entry.
;
ARPAGE 20
;
;
; ASSORTEDPARMS: Passes initialization parameters to TCPIP.  However,
;   use of this statement is being phased out.  Use the GLOBALCONFIG,
;   IPCONFIG, TCPCONFIG, and UDPCONFIG statements instead.
;
; GLOBALCONFIG: Provides settings for the entire TCP/IP stack
;
GLOBALCONFIG NOTCPIPSTATISTICS
;
; IPCONFIG: Provides settings for the IP layer of TCP/IP.
;
; Example IPCONFIG for single stack/single system:
;
IPCONFIG DATAGRAMFWD VARSUBNETTING SYSPLEXROUTING
;
; Example IPCONFIG for automatic activation of inter-stack dynamic XCF
;   and Same Host (IUTSAMEH) links
;
; IPCONFIG DYNAMICXCF 201.1.10.10 255.255.255.0 2
;
; KEEPALIVEOPTIONS: Specifies operating parameters of the TCP keep-alive
;   packets.  These same parameters can be specified on the TCPCONFIG
;   statement as well; use of the TCPCONFIG statement is recommended.
;
;
; SOMAXCONN: Specifies maximum length for the connection request queue
;   created by the socket call listen().
;
SOMAXCONN 10
;
;
; TCPCONFIG: Provides settings for the TCP layer of TCP/IP.
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to APF authorized or superuser applications.
;
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
TCPCONFIG RESTRICTLOWPORTS
;
;
```

```
; UDPCONFIG: Provides settings for the UDP layer of TCP/IP
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to APF authorized or superuser applications.
;
UDPCONFIG RESTRICTLOWPORTS
;
;
; ====================================================================
; Hardware definitions
; ====================================================================
;
; DEVICE: Defines name (and sometimes device number) for various types
;   of network devices
; LINK: Defines a network interface to be associated with a particular
;   device
;
;
; DEVICE and LINK for CTC devices
;
;   DEVICE  CTC1    CTC  D00  AUTORESTART
;   LINK    CTCD00  CTC  0  CTC1
;
; DEVICE and LINK for HYPERchannel A220 devices:
;
;   DEVICE  HCH1    HCH  E00  AUTORESTART
;   LINK    HCHE00  HCH  1  HCH1
;
; DEVICE and LINK for LAN Channel Station and OSA devices:
;   DEVICE: Defines name and hexadecimal device number for an IBM 8232
;     LAN channel station (LCS) device, and IBM 3172 Interconnect
;     Controller, an IBM 2216 Multiaccess Connector Model 400,
;     an IBM FDDI, Ethernet, or Token Ring OSA, or an IBM ATM OSA-2
;     in LAN emulation mode
;   LINK: Defines a network interface link associated with an LCS
;     device; may be for Ethernet Network, Token-Ring Network or
;     PC Network, or FDDI.
;
;   Example: LCS1 is a 3172 model 1 with a Token Ring and Ethernet
;    adapter
;
;     DEVICE  LCS1  LCS  BA0  AUTORESTART
;     LINK    TR1   IBMTR  0  LCS1
;     LINK    ETH1  ETHERNET  1  LCS1
;
;   Example: LCS2 is a 3172 model 2 with a FDDI adapter
;
;     DEVICE  LCS2   LCS  BE0  AUTORESTART
;     LINK    FDDI1  FDDI  0  LCS2
;
; DEVICE and LINK for MPCIPA QDIO Devices:
;
;   Example: MPCIPA1 is either an IBM OSA-Express Gigabit Ethernet
;    or QDIO Fast Ethernet adapter
;
;   DEVICE  MPCIPA1     MPCIPA  NONROUTER AUTORESTART
;   LINK    MPCIPALINK1 IPAQENET  MPCIPA1
;
;   Example: MPCIPA2 is either an IBM OSA-Express Gigabit Ethernet
;    or QDIO Fast Ethernet adapter, configured as the PRIMARY router
;
;   DEVICE  MPCIPA2     MPCIPA  PRIROUTER AUTORESTART
;   LINK    MPCIPALINK2 IPAQENET  MPCIPA2
;
; DEVICE and LINK for MPCPTP devices:
;
;   DEVICE  MPCPTP1    MPCPTP   AUTORESTART
;   LINK    MPCPTPLINK MPCPTP   MPCPTP1
```

```
;
; DEVICE and LINK for CLAW devices:
;
;   DEVICE  RS6K    CLAW  6B2  HOST  PSCA  NONE  26 26 AUTORESTART
;   LINK    IPLINK1 IP  0  RS6K
;
; DEVICE and LINK for SNA LU0 links:
;
;   DEVICE  SNALU0  SNAIUCV   SNALINK  LU000000  SNALINK  AUTORESTART
;   LINK    SNA1    SAMEHOST  1  SNALU0
;
; DEVICE and LINK for SNA LU 6.2 links:
;
;   DEVICE  SNALU621 SNALU62 SNAPROC AUTORESTART
;   LINK    SNA2     SAMEHOST  1  SNALU621
;
; DEVICE and LINK for X.25 NPSI connections:
;
;   DEVICE  X25DEV   X25NPSI  TCPIPX25  AUTORESTART
;   LINK    X25LINK  SAMEHOST  1  X25DEV
;
; DEVICE and LINK for 3745/46 Channel DLC Devices:
;
;   DEVICE  CDLC1    CDLC  C00 AUTORESTART
;   LINK    CDLCLINK CDLC  1  CDLC1
;
; DEVICE and LINK for MPC OSA Fast Ethernet Devices:
;
;   DEVICE  MENET1   MPCOSA   AUTORESTART
;   LINK    ENETLINK OSAENET 0 MENET1
;
; DEVICE and LINK for MPC OSA FDDI Devices:
;
;   DEVICE  MFDDI1   MPCOSA   AUTORESTART
;   LINK    FDDILINK OSAFDDI 0 MFDDI1
;
;
; ------------------------------------------------------------------------
; Virtual device definitions
; ------------------------------------------------------------------------
;
; DEVICE and LINK for Virtual Devices (VIPA):
;
;   DEVICE  VDEV1   VIRTUAL  0
;   LINK    VLINK1  VIRTUAL  0  VDEV1
;
; Dynamic Virtual Devices can be defined on this system.  This system
; can serve as backup for Dynamic Virtual Devices on other systems.
; A predefined range will allow Dynamic Virtual Devices to be defined
; by IOCTL or Bind requests.
;
; VIPADYNAMIC
;   Define two dynamic VIPAs on this stack:
;   VIPADEFINE 255.255.255.192 201.2.10.11 201.2.10.12
;
;   Define this stack as backup for these dynamic VIPAs on
;   other TCP/IP stacks:
;   VIPABACKUP 100             201.2.10.13 201.2.10.14
;   VIPABACKUP  80             201.2.10.21 201.2.10.22
;   VIPABACKUP  60             201.2.10.31 201.2.10.33
;   VIPABACKUP  40             201.2.10.32 201.2.10.34
;
;   VIPARANGE DEFINE 255.255.255.192 201.2.10.192
; ENDVIPADYNAMIC
;
; ------------------------------------------------------------------------
; ATM hardware definitions
```

```
; ----------------------------------------------------------------------
;
; ATMLIS: Describes characteristics of an ATM logical IP subnet (LIS).
;
; DEVICE and LINK for ATM devices: (See below)
;
; ATMPVC: Describes a permanent virtual circuit (PVC) to be used by an
;   ATM link.
;
; ATMARPSV: Designates the ATMARP server that will resolve ATMARP
;   requests for a logical IP subnet (LIS).
;
; ATMLIS   LIS1   9.67.100.0 255.255.255.0
; DEVICE   OSA1   ATM  PORTNAME  PORT1
; LINK     LINK1  ATM  OSA1  LIS LIS1
; ATMPVC   PVC1   LINK1
; ATMARPSV ARPSV1 LIS1 PVC PVC1
;
;
; ----------------------------------------------------------------------
; Other device statements
; ----------------------------------------------------------------------
;
; START: Starts a device that is currently stopped.
;
; START LCS1
; START LCS2
;
;
; TRANSLATE: Indicates a relationship between an internet address and
;   the network address on a specified link.
;
; TRANSLATE
;   9.67.43.110   FDDI    FF0000006702   FDDI1
;   9.37.84.49    HCH     FF0000005555   HCHE00
;
;
; ======================================================================
; HOME addresses
; ======================================================================
;
; HOME: Provides the list of home IP addresses and associated link names
;
;   - The LOOPBACK statement of 14.0.0.0 should only be used if the
;     installation has applications that require this old loopback
;     address.  The current stack uses 127.0.0.1 as the loopback
;     address.
;
; HOME
;  14.0.0.0       LOOPBACK
;  130.50.75.1    TR1
;  193.5.2.1      ETH1
;  9.67.43.110    FDDI1
;  193.7.2.1      SNA1
;  9.67.113.80    CTCD00
;  9.37.84.49     HCHE00
;  9.67.113.81    MPCIPALINK1
;  9.67.113.82    MPCPTPLINK
;  9.67.113.83    MPCIPALINK2
;  9.67.114.02    IPLINK1
;  9.67.43.03     SNA2
;  9.67.115.85    X25LINK
;  9.67.116.86    VLINK1
;  9.67.117.87    CDLCLINK
;  9.67.100.80    LINK1
;  9.37.112.13    ENETLINK
;  9.37.112.14    FDDILINK
```

```
;
;
; PRIMARYINTERFACE: Specifies which link is designated as the default
;   local host for use by the GETHOSTID() function.
;
;   - If PRIMARYINTERFACE is not specified, then the first link in
;     the HOME statement is the primary interface, as usual.
;
; PRIMARYINTERFACE TR1
;
;
; ====================================================================
; Routing configuration
; ====================================================================
; --------------------------------------------------------------------
; Static routing
; --------------------------------------------------------------------
;
; GATEWAY: Defines static routes to the IP route table.
;
; GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
;  Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
;
;  130.50       =         TR1       2000       0.0.255.0    0.0.75.0
;  193.5.2      =         ETH1      1500       0
;  9            =         FDDI1     4000       0.255.255.0  0.67.43.0
;  193.7.2.2    =         SNA1      2000       HOST
;
;
; Indirect Routes - Routes that are reachable through routers on my
;                   network.
;
;  Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
;
;  193.12.2 130.50.75.10  TR1       2000       0
;  10.5.6.4 193.5.2.10    ETH1      1500       HOST
;
;
; Default Route - All packets to an unknown destination are routed
;                 through this route.
;
;  Network  First Hop   Link Name Packet Size  Subnet Mask  Subnet Value
;
;  DEFAULTNET 9.67.43.99  FDDI1    DEFAULTSIZE  0
;
;
; --------------------------------------------------------------------
; Dynamic routing
; --------------------------------------------------------------------
;
; BSDROUTINGPARMS: Defines the characteristics of each link defined at
;   the host over which OROUTED will send routing information to
;   adjacent routers running the RIP protocl and which NCPROUTE will
;   send transport PDUs to client NCPs.
;
;   - OMPROUTE is the recommended routing daemon.  It does not use
;     BSDROUTINGPARMS.
;
;   - OROUTED users must define BSDROUTINGPARMS.
;
;   - Use of the GATEWAY statement (static routes) with the OMPROUTE
;     OROUTED routing daemons is not recommended.
;
; BSDROUTINGPARMS TRUE
```

```
;  Link name      MTU    Cost metric  Subnet Mask    Dest address
;   TR1          2000         0       255.255.255.0     0
;   ETH1         1500         0       255.255.255.0     0
;   FDDI1        4000         0       255.255.255.0     0
;   VLINK1    DEFAULTSIZE     0       255.255.255.0     0
;   CTCD00       65527        0       255.255.255.0  9.67.113.90
; ENDBSDROUTINGPARMS
;
;
; ====================================================================
; Application configuration
; ====================================================================
;
; AUTOLOG: Supplies TCPIP with the procedure names to start and the
;  timeout value to use for a hung procedure during AUTOLOG.
;
; AUTOLOG 5
;  FTPD JOBNAME FTPD1       ; FTP Server
;  LPSERVE                  ; LPD Server
;  NAMED                    ; Domain Name Server
;  NCPROUT                  ; NCPROUTE Server
;  OROUTED                  ; OROUTED Server
;  OSNMPD                   ; SNMP Agent Server
;  PORTMAP                  ; Portmap Server (SUN 3.9)
;  PORTMAP JOBNAME PORTMAP1 ; USS Portmap Server (SUN 4.0)
;  RXSERVE                  ; Remote Execution Server
;  SMTP                     ; SMTP Server
;  SNMPQE                   ; SNMP Client
;  TCPIPX25                 ; X25 Server
; ENDAUTOLOG
;
;
; PORT: Reserves a port for specified job names
;
;   - A port that is not reserved in this list can be used by any user.
;     If you have TCP/IP hosts in your network that reserve ports
;     in the range 1-1023 for privileged applications, you should
;     reserve them here to prevent users from using them.
;     The RESTRICTLOWPORTS option on TCPCONFIG and UDPCONFIG will also
;     prevent unauthorized applications from accessing unreserved
;     ports in the 1-1023 range.
;
;   - A PORT statement with the optional keyword SAF followed by a
;     1-8 character name can be used to reserve a PORT and control
;     access to the PORT with a security product such as RACF.
;     For port access control, the full resource name for the security
;     product authorization check is constructed as follows:
;     EZB.PORTACCESS.sysname.tcpname.safname
;     where:
;       EZB.PORTACCESS is a constant
;       sysname is the MVS system name (substitute your sysname)
;       tcpname is the TCPIP jobname (substitute your jobname)
;       safname is the 1-8 character name following the SAF keyword
;
;     When PORT access control is used, the TCP/IP application
;     requiring access to the reserved PORT must be running under a
;     USERID that is authorized to the resource. The resources
;     are defined in the SERVAUTH class.
;
;     For an example of how the SAF keyword can be used to enhance
;     security, see the definition below for the FTP data PORT 20
;     with the SAF keyword. This definition reserves TCP PORT 20 for
;     any jobname (the *) but requires that the FTP user be permitted
;     by the security product to the resource:
;     EZB.PORTACCESS.sysname.tcpname.FTPDATA in the SERVAUTH class.
;
;   - The BIND keyword is used to force a generic server (one that
```

```
;     binds to INADDR_ANY) to bind to the specific IP address that
;     is specified following the BIND keyword. This capability could
;     be used, for example, to allow OE telnet and telnet 3270 servers
;     to both bind to TCP port 23. The IP address that follows bind
;     must be in IPv4 dotted decimal format and may be any valid
;     address for the host including VIPA and dynamic VIPA addresses.
;
;   The special jobname of OMVS indicates that the PORT is reserved
;   for any application with the exception of those that use the Pascal
;   API.
;
;   The special jobname of * indicates that the PORT is reserved
;   for any application, including Pascal API socket applications.
;
;   The special jobname of RESERVED indicates that the PORT is
;   blocked. It will not be available to any application.
;
;   The special jobname of INTCLIEN indicates that the PORT is
;   reserved for internal stack use.
;
;
PORT
     7 UDP MISCSERV            ; Miscellaneous Server - echo
     7 TCP MISCSERV            ; Miscellaneous Server - echo
     9 UDP MISCSERV            ; Miscellaneous Server - discard
     9 TCP MISCSERV            ; Miscellaneous Server - discard
    19 UDP MISCSERV            ; Miscellaneous Server - chargen
    19 TCP MISCSERV            ; Miscellaneous Server - chargen
    20 TCP *  NOAUTOLOG        ; FTP Server
;   20 TCP *  NOAUTOLOG SAF FTPDATA ; FTP Server
    21 TCP FTPD1               ; FTP Server
    23 TCP INTCLIEN            ; Telnet 3270 Server
;   23 TCP INETD1 BIND 9.67.113.3 ; OE telnet server
    25 TCP SMTP                ; SMTP Server
    53 TCP NAMED               ; Domain Name Server
    53 UDP NAMED               ; Domain Name Server
   111 TCP PORTMAP             ; Portmap Server (SUN 3.9)
   111 UDP PORTMAP             ; Portmap Server (SUN 3.9)
;  111 TCP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
;  111 UDP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
   135 UDP LLBD                ; NCS Location Broker
   161 UDP OSNMPD              ; SNMP Agent
   162 UDP SNMPQE              ; SNMP Query Engine
   512 TCP RXSERVE             ; Remote Execution Server
   514 TCP RXSERVE             ; Remote Execution Server
;  512 TCP * SAF OREXECD       ; OE Remote Execution Server
;  514 TCP * SAF ORSHELLD      ; OE Remote Shell Server
   515 TCP LPSERVE             ; LPD Server
   520 UDP OROUTED             ; OROUTED Server
   580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERB             ; Kerberos
   750 UDP MVSKERB             ; Kerberos
   751 TCP ADM@SRV             ; Kerberos Admin Server
   751 UDP ADM@SRV             ; Kerberos Admin Server
  3000 TCP CICSTCP             ; CICS Socket
;
;
; PORTRANGE: Reserves a range of ports for specified jobnames.
;
;   In a common INET (CINET) environment, the port range indicated by
;   the INADDRANYPORT and INADDRANYCOUNT in your BPXPRMxx parmlib member
;   should be reserved for OMVS.
;
;   The special jobname of OMVS indicates that the PORTRANGE is reserved
;   for ANY OE socket application.
;
;   The special jobname of * indicates that the PORTRANGE is reserved
```

```
;   for any socket application, including Pascal API socket
;   applications.
;
;   The special jobname of RESERVED indicates that the PORTRANGE is
;   blocked. It will not be available to any application.
;
;   The SAF keyword is used to restrict access to the PORTRANGE to
;   authorized users. See the use of SAF on the PORT statement above.
;
;
;   PORTRANGE 4000 1000 TCP OMVS
;   PORTRANGE 4000 1000 UDP OMVS
;   PORTRANGE 2000 3000 TCP RESERVED
;   PORTRANGE 5000 6000 TCP * SAF RANGE1
;
; SACONFIG: Configures the TCP/IP SNMP subagent
;
SACONFIG ENABLED COMMUNITY public AGENT 161
;
;
; ---------------------------------------------------------------------
; Configure Telnet
; ---------------------------------------------------------------------
;
; TELNETPARMS: Configure the Telnet Server
;
; - TN3270(E) server port 23 options
;
TelnetParms
  Port 23                       ; Port number 23 (std.)
  CodePage ISO8859-1 IBM-1047   ; Linemode ASCII, EBCDIC code pages
  Inactive 0                    ; Let connections stay around
  PrtInactive 0                 ; Let connections stay around
  TimeMark 600
  ScanInterval 120
  SMFinit std
  SMFterm std
  WLMClusterName
    TN3270E
  EndWLMClusterName
EndTelnetParms
;
; TelnetParms
;   Secureport 992 Keyring HFS /tmp/telnet.kdb
; EndTelnetParms
;
; BEGINVTAM: Defines the VTAM parameters required for the Telnet server.
;
BeginVTAM
  Port 23 ; 992
  ; Define logon mode tables to be the defaults shipped with the
  ; latest level of VTAM
  TELNETDEVICE 3278-3-E NSX32703 ; 32 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3279-3-E NSX32703 ; 32 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3278-4-E NSX32704 ; 48 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3279-4-E NSX32704 ; 48 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3278-5-E NSX32705 ; 132 column screen-
                                 ; default of NSX32702 is 80
  TELNETDEVICE 3279-5-E NSX32705 ; 132 column screen -
                                 ; default of NSX32702 is 80

  ; Define the LUs to be used for general users.
  DEFAULTLUS
```

```
    TCPABC01..TCPABC99
  ENDDEFAULTLUS

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

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

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

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

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

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

  ALLOWAPPL *       ; Allow all applications that have not been
                    ; previously specified to be accessed.

  RESTRICTAPPL IMS ; Only 3 users can use IMS.
    USER USER1      ; Allow user1 access.
      LU TCPIMS01   ; Assign USER1 LU TCPIMS01.
    USER USER2      ; Allow user2 access from the default LU pool.
    USER USER3      ; Allow user3 access from 3 Telnet sessions,
                    ; each with a different reserved LU.
      LU TCPIMS31 LU TCPIMS32 LU TCPIMS33

  ;   Map Telnet sessions from IP address 130.50.10.1 to display the
  ;   USSMSG10 screen from USS table USSAPC.

; USSTCP USSAPC 130.50.10.1

  ;   Map Telnet sessions from the SNA1 link to display the USSMSG10
  ;   screen from USS table USSCBA.

; USSTCP USSCBA SNA1

; LUGROUP  LUGRP1
;       TCPM0001..TCPM0999
;       TCPM1001
; ENDLUGROUP

; LUGROUP LUGRP2
;       TCPM2001  TCPM2003 TCPM2004
;       TCPM0AAA..TCPM0ZZZ
; ENDLUGROUP

  ; Define groups of host names
; HNGROUP HNGRP1
;      TEST1.TCP.RALEIGH.IBM.COM
;      TEST2.TCP.RALEIGH.IBM.COM
;      *.*.RALEIGH.IBM.COM
; ENDHNGROUP

; HNGROUP HNGRPALL
;      **.COM
; ENDHNGROUP

  ; Map LUs to groups for host names
```

```
; LUMAP LUGRP1 HNGRP1
; LUMAP LUGRP2 HNGRPALL
; LUMAP TCPM5000  SPECIAL.TCP.RALEIGH.IBM.COM
EndVTAM
;
;
; ---------------------------------------------------------------------
; Configure Network Access Control
; ---------------------------------------------------------------------
;     Network access contol can be used to restrict the destinations
;     that TCP/IP users are allowed to send to. The NETACCESS
;     group contains a list of IP destinations that may be subnetworks
;     or specific hosts. The subnetwork mask can be specified as a
;     number of significant bits or in dotted decimal notation.
;
;     A 1-8 character name follows the IP address and subnet mask and
;     is used as the right-most qualifier in the security product
;     resource name.
;
;     For network access control, the full resource name for the
;     security product authorization check is constructed as follows:
;
;     EZB.NETACCESS.sysname.tcpname.resname
;     where:
;       EZB.NETACCESS is a constant
;       sysname is the MVS system name (substitute your sysname)
;       tcpname is the TCPIP jobname (substitute your jobname)
;       resname is the 1-8 character name following the subnet mask.
;
;     When network access control is used, the TCP/IP application
;     requiring access to the restricted subnet or host must be running
;     under a USERID that is authorized to the resource. The resources
;     are defined in the SERVAUTH class. See the EZARACF sample for
;     examples of the RACF definitions.
;
;NETACCESS
;192.168.0.0/16            SUBNET1     ; Subnet address
;192.168.113.19/32         HOST1       ; Specific host address
;192.168.113.0 255.255.255.0   SUBNET2 ; Subnet address
;192.168.112.0 255.255.248.0   SUBNET3 ; Subnet address
;DEFAULT 0                 DEFZONE     ; Optional Default security zone
;ENDNETACCESS
;
;
;
; ====================================================================
; Diagnostic data statements
; ====================================================================
;
; - For optimum performance, use of tracing should be limited to when
;   required for problem analysis.
;
; ITRACE: Controls TCP/IP run-time tracing
;
; ITRACE ON CONFIG 1
; ITRACE OFF SUBAGENT
;
;
; PKTTRACE: Controls the packet trace facility in TCP/IP.
;
; PKTTRACE ABBREV=200 LINKNAME=TR1 PROT=ICMP IP=*
;   SRCPORT=5000 DESTPORT=161
;
;
; SMFCONFIG: Provides SMF logging for Telnet, FTP, TCP API and TCP
;   stack activity.
;
```

```
;   - The SMF record type for TCP/IP records is 118.
;
; SMFCONFIG TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
;
;
; SMFPARMS: Logs the use of TCP by applications using SMF log records.
;   However, use of the SMFCONFIG statement is recommended instead.
;
;
; ====================================================================
; Other statements
; ====================================================================
;
; DELETE: Removes an ATMARPSV, ATMLIS, ATMPVC, device, link, port or
;   portrange.  This statement is typically done via an obey file, not
;   in an initial profile.
;
; STOP: Stops a device.  If used, this statement is typically put in
;   an obey file, not in an initial profile.
;
; INCLUDE: Causes another data set that contains profile configuration
;   statements to be included at this point.
;
;
; --------------------------------------------------------------------
```

The following section explains the grouping of statements shown in Figure 14 on page 79.

DEVICE and LINK

Use DEVICE and LINK statements to define each network interface to the TCPIP address space. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details about the various network interfaces supported by TCP/IP.

ATM Use the ATM DEVICE and LINK statements to define connectivity to an ATM network. These statements allow for connectivity in either ATM native mode over an ATM virtual circuit (VC) or in ATM LAN Emulation mode.

For ATM native mode, the VC can be either a permanent virtual circuit (PVC) or a switched virtual circuit (SVC). To define a PVC, use the ATMPVC statement. To define SVCs, use the ATMLIS statement to define the ATM logical IP subnet (LIS). Also, for SVCs, use the ATMARPSV statement to define the ATMARP server that will resolve ATMARP requests within the LIS. For ATM LAN emulation mode, the ATM DEVICE and LINK definitions allow you to retrieve SNMP network management data for the device. In this mode, you need to define the device as an LCS.

CDLC The DEVICE CDLC describes the interface between the TCP/IP address space and the 3745/46 devices used.

CLAW Use CLAW DEVICE for RISC System/6000® and SP2®. The DEVICE statement has a parameter that must match the HOST name as specified of the RISC/6000 along with buffer number and size specifications.

CTC Use the CTC DEVICE and LINK statements to define connectivity to another z/OS using channel-to-channel.

HYPERchannel A220 DEVICE and LINK

Use the HCH DEVICE and LINK statements to define connectivity via the HYPERchannel A220 adapter.

LAN Channel Station (LCS) DEVICE and LINK

Use the LCS DEVICE and LINK statements to define connectivity to a token-ring, FDDI, or Ethernet LAN. LCS devices can have more than one adapter. Therefore, you can have more than one LINK statement for an LCS DEVICE statement. The DEVICE statement does have an optional NETMAN parameter to signify that the 3172 supports the IBM Enterprise-specific MIB variables. For token-ring adapters, the LINK statement has additional parameters describing MAC address formats and ARP broadcasts.

In configurations where multiple LCS and/or MPCIPA links onto the same LAN are defined, if the interface targeted by the ARP Request is inactive, one of the other active interfaces on the LAN will automatically take over responsibility for answering ARPs on behalf of the inactive interface. In this way, fault tolerance is achievable on the LAN without requiring a dynamic routing protocol.

TCP/IP supports ARP for VIPAs. In a flat network (one in which traffic flows directly between two endpoints without an intermediate router) using static routing with multiple interfaces onto the same LAN, you can achieve fault tolerance by defining a VIPA in the same subnet as the physical interfaces on the LAN. If a static route specifies a VIPA as the next hop IP address, the host or router will send an ARP for the VIPA. TCP/IP will reply to the ARP with the MAC address of one of the active physical interfaces on that LAN.

MPCOSA

The MPCOSA DEVICE statements define the MPC OSA Ethernet and FDDI devices.

MPCIPA

Use the MPCIPA DEVICE and LINK statements to define LAN connectivity via OSA-Express using the Queued Direct I/O (QDIO) interface. The MPCIPA device name must be the PORT name of the TRLE definition of the QDIO interface as described in *z/OS Communications Server: SNA Resource Definition Reference*. Device specifications for the type of IP routing supported are also specified on the MPCIPA DEVICE statement. These are also described in *z/OS Communications Server: SNA Resource Definition Reference*.

In configurations where multiple LCS and/or MPCIPA links onto the same LAN are defined, if the interface targeted by the ARP Request is inactive, one of the other active interfaces on the LAN will automatically take over responsibility for answering ARPs on behalf of the inactive interface. In this way, fault tolerance is achievable on the LAN without requiring a dynamic routing protocol.

TCP/IP supports ARP for VIPAs. In a flat network (one in which traffic flows directly between two endpoints without an intermediate router) using static routing with multiple interfaces onto the same LAN, you can achieve fault tolerance by defining a VIPA in the same subnet as the physical interfaces on the LAN. If a static route specifies a VIPA as the next hop IP address, the host or router will send an ARP for the VIPA. TCP/IP will reply to the ARP with the MAC address of one of the active physical interfaces on that LAN.

MPCPTP

MPCPTP can be used to define any of the following:

- A connection to another host over a series of CTCs (in this case, the device name must be the name of a VTAM TRLE)
- An XCF connection to another TCP/IP in the same z/OS sysplex. For an XCF connection, the device name must be the cp name of the target VTAM on the other side of the XCF connection, and the VTAM ISTLSXCF major node must be active to start the device.
- An IUTSAMEH connection (with no need for any I/O devices) to another TCP/IP on the same z/OS system or to VTAM for Enterprise Extender. For an IUTSAMEH connection, the device name must be the reserved name IUTSAMEH. VTAM automatically activates the IUTSAMEH TRLE.

Finally, the MPCPTP LINK statement has a NOCHECKSUM option to eliminate inbound checksum processing on the interface. Only specify NOCHECKSUM for reliable, single-hop MPCPTP links.Use the IPCONFIG DYNAMICXCF statement to cause TCP/IP to automatically define and activate XCF connectivity between each pair of TCP/IP stacks in the same sysplex and IUTSAMEH connectivity between multiple TCP/IP stacks on the same z/OS.

X.25 The DEVICE X25 DEV defines the interface between the TCP/IP address space and the address space of the X.25 NPSI server.

VIPA and VIPADYNAMIC

Virtual IP Addresses (VIPA) are used to define virtual devices to the TCP/IP address space. There are two types of VIPAs:

- Static
- Dynamic

The static virtual device requires DEVICE and LINK statements to define a device that is always started, can never be stopped, can be known within the network, yet requires no physical adapters. It is very useful to define VIPAs so that if a physical adapter loses its connection to the network, application traffic using the failed physical adapter can be rerouted over another interface to the network. To the network, the VIPA address appears to be one hop away from the TCP/IP address spaces. The network sends and receives datagrams to and from the physical interfaces to get to the VIPA address. For more information about VIPA, see “Chapter 3. Virtual IP Addressing” on page 109.

HOME HOME lists the IP addresses and their associated LINK adapter. The first HOME statement within a configuration data set replaces the existing HOME list. If subsequent HOME statements are found within a configuration data set, add entries to the list.

Note: If IPCONFIG SOURCEVIPA is specified, the order of the VIPA addresses in the HOME list is important. The source address used will be the preceding VIPA address instead of the physical adapter used to send the datagram. If no VIPA precedes the physical adapter in the HOME list, the physical adapter IP address is used as the source address. Refer to *z/OS Communications Server: IP Configuration Reference* for precautions when either the VIPA address or a physical adapter used as a source for the VIPA has a IP address that is the network address.

PRIMARYINTERFACE
Use PRIMARYINTERFACE to specify which link should be designated as the default local host for use by the GETHOSTID() function. If PRIMARYINTERFACE is not used, the first IP address in the HOME list becomes the default local host address.

SMFCONFIG
Use the SMFCONFIG statement to provide SMF logging for Telnet, FTP, TCP, API, and stack activity. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the SMFCONFIG statement.

SNAIUCV
Use SNAIUCV DEVICE to specify the interface to use for SNA LU0 traffic to the SNALINK started procedures. For example, use this to define the interface between the TCP/IP address space and the SNALINK address space that is using a 3745 running NCPRoute. Similarly, the DEVICE SNA LU62 statement defines the interface between the TCP/IP address space and the address space using SNA LU62. Refer to *z/OS Communications Server: IP Configuration Reference* for information about how to define multiple LU6.2 connections within the same TCP/IP address space.

START
Use START to activate a device.

TRANSLATE
Use TRANSLATE to indicate which LINK has specified network addresses for use as a static ARP table. Like HOME, the first TRANSLATE statement in a configuration data set replaces the entire static ARP table entry. Subsequent TRANSLATE statements add to the table. If you are using OSPF routing (OMPROUTE), see "Chapter 4. Routing" on page 143 for more information about requirements for the TRANSLATE statement.

After an interface has a DEVICE, LINK, and HOME statement, it can be started via the START device statement or the VARY TCPIP,,START command.

Devices that Support ARP Offload

Certain devices provide an ARP offload function that offloads all ARP processing to the adapter. The function provided by the adapter impacts the ability of TCP/IP to display ARP cache information or ARP counter statistics for these devices.

The following devices provide an ARP offload function and provide ARP cache data or ARP counters to TCP/IP.

- MPCIPA OSA-Express Gigabit Ethernet (minimum required microcode level [MCL] 401)
- MPCIPA (OSA-Express Fast Ethernet)

Note: If multiple TCP/IP instances are sharing the device, the ARP data will represent all TCP/IP instances using the device. This information is provided to TCP/IP every 30 seconds from the device.

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

- MPCOSA (OSA-2 Fast Ethernet, FDDI)
- MPCIPA (OSA-Express Gigabit Ethernet) with a microcode level earlier than [MCL] 401

Setting Up Reserved Port Number Definitions in PROFILE.TCPIP

The following example shows how to set up reserved port number definitions in PROFILE.TCPIP. For more detailed information about any of these statements, refer to *z/OS Communications Server: IP Configuration Reference*.

Figure 15. Example of Reserved Port Number Definitions

```
;
; ====================================================================
; Application configuration
; ====================================================================
;
; AUTOLOG: Supplies TCPIP with the procedure names to start and the
;  timeout value to use for a hung procedure during AUTOLOG.
;
; AUTOLOG 5
;  FTPD JOBNAME FTPD1       ; FTP Server
;  LPSERVE                  ; LPD Server
;  NAMED                    ; Domain Name Server
;  NCPROUT                  ; NCPROUTE Server
;  OROUTED                  ; OROUTED Server
;  OSNMPD                   ; SNMP Agent Server
;  PORTMAP                  ; Portmap Server (SUN 3.9)
;  PORTMAP JOBNAME PORTMAP1 ; USS Portmap Server (SUN 4.0)
;  RXSERVE                  ; Remote Execution Server
;  SMTP                     ; SMTP Server
;  SNMPQE                   ; SNMP Client
;  TCPIPX25                 ; X25 Server
; ENDAUTOLOG
;
;
; PORT: Reserves a port for specified job names
;
;   - A port that is not reserved in this list can be used by any user.
;     If you have TCP/IP hosts in your network that reserve ports
;     in the range 1-1023 for privileged applications, you should
;     reserve them here to prevent users from using them.
;     The RESTRICTLOWPORTS option on TCPCONFIG and UDPCONFIG will also
;     prevent unauthorized applications from accessing unreserved
;     ports in the 1-1023 range.
;
;   - A PORT statement with the optional keyword SAF followed by a
;     1-8 character name can be used to reserve a PORT and control
;     access to the PORT with a security product such as RACF.
;     For port access control, the full resource name for the security
;     product authorization check is constructed as follows:
;     EZB.PORTACCESS.sysname.tcpname.safname
;     where:
;       EZB.PORTACCESS is a constant
;       sysname is the MVS system name (substitute your sysname)
;       tcpname is the TCPIP jobname (substitute your jobname)
;       safname is the 1-8 character name following the SAF keyword
;
;     When PORT access control is used, the TCP/IP application
;     requiring access to the reserved PORT must be running under a
;     USERID that is authorized to the resource. The resources
;     are defined in the SERVAUTH class.
;
;     For an example of how the SAF keyword can be used to enhance
;     security, see the definition below for the FTP data PORT 20
;     with the SAF keyword. This definition reserves TCP PORT 20 for
;     any jobname (the *) but requires that the FTP user be permitted
;     by the security product to the resource:
;     EZB.PORTACCESS.sysname.tcpname.FTPDATA in the SERVAUTH class.
;
```

```
;   - The BIND keyword is used to force a generic server (one that
;     binds to INADDR_ANY) to bind to the specific IP address that
;     is specified following the BIND keyword. This capability could
;     be used, for example, to allow OE telnet and telnet 3270 servers
;     to both bind to TCP port 23. The IP address that follows bind
;     must be in IPv4 dotted decimal format and may be any valid
;     address for the host including VIPA and dynamic VIPA addresses.
;
;   The special jobname of OMVS indicates that the PORT is reserved
;   for any application with the exception of those that use the Pascal
;   API.
;
;   The special jobname of * indicates that the PORT is reserved
;   for any application, including Pascal API socket applications.
;
;   The special jobname of RESERVED indicates that the PORT is
;   blocked. It will not be available to any application.
;
;   The special jobname of INTCLIEN indicates that the PORT is
;   reserved for internal stack use.
;
;
PORT
     7 UDP MISCSERV            ; Miscellaneous Server - echo
     7 TCP MISCSERV            ; Miscellaneous Server - echo
     9 UDP MISCSERV            ; Miscellaneous Server - discard
     9 TCP MISCSERV            ; Miscellaneous Server - discard
    19 UDP MISCSERV            ; Miscellaneous Server - chargen
    19 TCP MISCSERV            ; Miscellaneous Server - chargen
    20 TCP *  NOAUTOLOG        ; FTP Server
;   20 TCP *  NOAUTOLOG SAF FTPDATA ; FTP Server
    21 TCP FTPD1               ; FTP Server
    23 TCP INTCLIEN            ; Telnet 3270 Server
;   23 TCP INETD1 BIND 9.67.113.3 ; OE telnet server
    25 TCP SMTP                ; SMTP Server
    53 TCP NAMED               ; Domain Name Server
    53 UDP NAMED               ; Domain Name Server
   111 TCP PORTMAP             ; Portmap Server (SUN 3.9)
   111 UDP PORTMAP             ; Portmap Server (SUN 3.9)
;  111 TCP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
;  111 UDP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
   135 UDP LLBD                ; NCS Location Broker
   161 UDP OSNMPD              ; SNMP Agent
   162 UDP SNMPQE              ; SNMP Query Engine
   512 TCP RXSERVE             ; Remote Execution Server
   514 TCP RXSERVE             ; Remote Execution Server
;  512 TCP * SAF OREXECD       ; OE Remote Execution Server
;  514 TCP * SAF ORSHELLD      ; OE Remote Shell Server
   515 TCP LPSERVE             ; LPD Server
   520 UDP OROUTED             ; OROUTED Server
   580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERB             ; Kerberos
   750 UDP MVSKERB             ; Kerberos
   751 TCP ADM@SRV             ; Kerberos Admin Server
   751 UDP ADM@SRV             ; Kerberos Admin Server
  3000 TCP CICSTCP             ; CICS Socket
;
;
; PORTRANGE: Reserves a range of ports for specified jobnames.
;
;   In a common INET (CINET) environment, the port range indicated by
;   the INADDRANYPORT and INADDRANYCOUNT in your BPXPRMxx parmlib member
;   should be reserved for OMVS.
;
;   The special jobname of OMVS indicates that the PORTRANGE is reserved
;   for ANY OE socket application.
;
```

```
;   The special jobname of * indicates that the PORTRANGE is reserved
;   for any socket application, including Pascal API socket
;   applications.
;
;   The special jobname of RESERVED indicates that the PORTRANGE is
;   blocked. It will not be available to any application.
;
;   The SAF keyword is used to restrict access to the PORTRANGE to
;   authorized users. See the use of SAF on the PORT statement above.
;
;
;   PORTRANGE 4000 1000 TCP OMVS
;   PORTRANGE 4000 1000 UDP OMVS
;   PORTRANGE 2000 3000 TCP RESERVED
;   PORTRANGE 5000 6000 TCP * SAF RANGE1
;
; SACONFIG: Configures the TCP/IP SNMP subagent
;
SACONFIG ENABLED COMMUNITY public AGENT 161
```

The following explains the statements shown in Figure 15 on page 94.

AUTOLOG

Use AUTOLOG to list the procedure names that should start when the TCPIP address space starts. It is also used to supply a timeout value for detecting hung procedures. The timeout value specifies how frequently the autolog task should check for hung procedures. A hung procedure is active to MVS, but is not listening on the socket that is reserved for it via the PORT statement. When AUTOLOG detects a hung task, it uses the MVS CANCEL command to terminate the procedure and then uses the MVS START command to start another copy.

The AUTOLOG statement shown in Figure 15 on page 94 has a timeout value of five minutes. This means that the autolog task checks for hung tasks in five minute intervals.

In the first AUTOLOG statement, the FTP Server, shows FTPD JOBNAME FTPD1. This means when the TCPIP address space starts, the FTPD procedure will be started via the MVS START FTPD command. Because FTPD forks a child process that actually listens on PORT 21 (see the PORT statement in this section), the autolog task verifies that FTPD1 is listening on port 21.

Similarly, when the TCPIP address space starts, the autolog task starts the remaining 10 tasks.

Unless the tasks in the AUTOLOG list are in the PORT reservation list, the autolog task does not check for hung tasks every five minutes. Also at startup time, those procedures that are not on the PORT list are first canceled and then started. This occurs because the procedure might have been running from a previous TCP/IP address space and would need to be started and stopped to start listening when the new stack starts.

Note: If you run multiple TCP/IP address spaces, ensure that the second address space AUTOLOG list does not cancel the procedures of the first. In those cases, an installation might require different procedure names for the servers for each address space. For more information about multiple stacks, see “Port Management Overview” on page 38.

For those procedures that require parameters to be used on the MVS START command, there is a PARMSTRING option. For more information, refer to *z/OS Communications Server: IP Configuration Reference*.

PORT Use PORT to reserve ports for different jobs. This prevents a rogue application from taking port 21, which is needed by FTP. For each port entry, the port number, protocol, and procedure name are specified. The first port entry shows port 7 UDP reserved for the miscellaneous echo server for procedure MISCSERV. Similarly, port 7 of TCP is also reserved for the same server. In this example, six ports are reserved for the miscellaneous server.

INTCLIEN is an INTernal CLIENt to the TCPIP address space (the TN3270 Telnet server), and it runs continuously. See “Chapter 6. Accessing Remote Hosts Using Telnet” on page 225 for more information about INTCLIEN.

NOAUTOLOG can be specified, as in the port 20 TCP * in Figure 15 on page 94. In this way, the port is reserved for an OMVS forked task so that the FTP server can fork tasks to port 20 as each FTP user logs in.

Use the DELAYACKS and NODELAYACKS options to allow an installation to delay their acknowledgments so they can be combined with data to be sent to foreign hosts. Unless a performance reason is needed, NODELAYACKS should be used to immediately send acknowledgments.

Use OPTMSS for TCP ports to specify the optimal maximum segment size function that should be used to calculate the MSS.

Use SHAREPORT when reserving a port to be shared across multiple TCP listeners. This is not valid for UDP.

Typically, reserving a port for a specific job name is sufficient. If the port must instead be reserved for a specific user ID or a set of user IDs, use the SAF keyword to specify the name of a SAF resource to be associated with the port. The user ID associated with the application that attempts to bind to the port must be permitted to the SAF resource.

The BIND keyword is used to force a generic server (one that binds to INADDR_ANY) to bind to the specific IP address that is specified following the BIND keyword. This capability could be used, for example, to allow OE Telnet and Telnet 3270 servers to both bind to TCP port 23 on different IP addresses. The IP address that follows bind must be in IPv4 dotted decimal format and can be any valid address for the host including VIPA and dynamic VIPA addresses. For multiple servers to bind to the same port with this function, the IP address for each server must be unique.

RESERVED indicates that the port is not available for use by any user.

PORTRANGE
PORTRANGE is a statement used to reserve a range of ports for specified job names.

SACONFIG
SACONFIG is the statement used to configure the information about the SNMP subagent. Use SACONFIG to specify the following:

- SNMP community string
- OSA/SF port number
- Agent port
- OSA management support
- Whether or not the SNMP agent can perform SNMP sets

Setting Up SAF Server Access Authorization (SERVAUTH) Optional

The TCP/IP address space uses the SERVAUTH Security Access Facility (SAF) class to protect TCP/IP resources from unauthorized access. The use of SERVAUTH is optional and is available in degrees so that installations can pick and choose the access needed. Installations can choose to use one, all, or none of the protections provided by SERVAUTH. The customization described in this section is completely optional. A template of the commands discussed in this section and all other SAF commands appears in SEZAINST(EZARACF).

Stack Access

z/OS CS allows an installation to prevent users from getting access to the TCP/IP stack through a TCP or UDP socket. For example, users can be restricted from accessing information using NETSTAT, TN3270 Telnet, or FTP.

To minimize performance impacts, the stack access check is only performed on the socket() call. Additionally, the security checking will be bypassed for socket() calls originating in the TCP/IP, UNIX System Services or VTAM address spaces. If SERVAUTH stack access profiles are enabled, all TCP/IP code must be running under a user ID that is permitted to the SAF resource ″EZB.STACKACCESS.sysname.tcpipname″ where ″sysname″ is the name of the MVS image in the sysplex and the last qualifier is the TCP/IP job name. If there are multiple stacks in the z/OS system, each stack can be enabled for stack access or not. This means that in a common INET environment, if some stacks are protected and some are not, a user may gain access to the network via the non-protected stack. Examples of non-protected stacks include AnyNet® Sockets Over SNA and non-IBM stacks. Also, if the system administrator does not define the SAF resource to protect the stack, stack access control is not performed and all users have access to the stack.

NetAccess

z/OS CS allows an installation to disallow the access of certain users to certain networks. The customer essentially classifies IP networks into security zones as viewed by the TCP/IP host, in which a network is considered to have a certain level of security sensitivity. Firewall products disallow communication between hosts based on interface IP addresses, but usually cannot differentiate between users on a host. Proxy servers can provide similar functions at the user level but this tends to necessitate complex configurations.

Therefore, z/OS CS has the ability to disallow communication between a particular user or group of users on the z/OS host and a particular network by specifically disallowing the users the ability to send IP data packets to the destination network or host. Note that only sends are restricted and this should provide network access without overly burdening performance by checking all incoming IP data packets.

Network access could be quite useful if only a selected group of users should have access to the Internet. The following example allows only USER1 access to the Internet. The following are the RACF commands entered from TSO:

```
NETACCESS
; NETWORK                  SAF
  9.0.0.0/8                IBM
  DEFAULT 0                INTERNET
  ENDNETACCESS

SETROPTS CLASSACT(SERVAUTH) REFRESH
RDEFINE SERVAUTH (EZB.NETACCESS.MVSVIC97.TCPCS.IBM ) UACC(READ)
RDEFINE SERVAUTH (EZB.NETACCESS.MVSVIC97.TCPCS.INTERNET ) UACC(NONE)
PERMIT EZB.NETACCESS.MVSVIC97.TCPCS.INTERNET  ACCESS(READ) CLASS(SERVAUTH) ID(USER1)
```

```
SETROPTS CLASSACT(SERVAUTH) REFRESH RACLIST(SERVAUTH)

D TCPIP,,Netstat,ACCess,NETWork
EZZ2500I NETSTAT CS V2R10 TCPCS
NETWORK ACCESS INFORMATION
NETWORK PREFIX  ADDRESS MASK    SAF NAME
DEFAULT         0.0.0.0         INTERNET
9.0.0.0         255.0.0.0       IBM
2 OF 2 RECORDS DISPLAYED
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information on the NETACCESS statement.

PortAccess

Port access allows a z/OS CS installation the ability to SAF-authorize port numbers to applications. By so doing, rogue applications cannot assume the function of an application normally using a defined port number. (Ports can also be reserved for a specific job name.) A SAF resource name is coded on the PORT or PORTRANGE statement. Any application attempting to use the specified port number requires SAF access to the resource. The SERVAUTH resource name is the string EZB.PORTACCESS, followed by the system name, then the stack name, then the resource name specified by the SAF keyword.

For example, a port reserved with PORT 600 TCP * SAF MYAPP (in PROFILE.TCPIP) on the system named MVSSYS1 and TCP/IP stack named TCPCS would have a profile of the following form:

```
EZB.PORTACCESS.MVSSYS1.TCPCS.MYAPP
```

Use of this form of port reservation could be used when a reserved low port must be accessed by many potential users via a client program that is not APF-authorized. All users needing the ability to run this program could be permitted to the RACF resource EZB.PORTACCESS.MVSSYS1.TCPCS.MYAPP.

Note: The resource name can be any valid character string, but is limited to eight characters. The SAF resource name parameter can permit multiple users access to the protected port. The stack, however, only allows one application to actually bind to the TCP port at a time (unless SHAREPORT or BIND is used). Refer to *z/OS Communications Server: IP Configuration Reference* for more information on the PORT and PORTRANGE statements.

The following example shows the operator command to display the port reservation for the examples above:

```
D TCPIP,,NETSTAT,PORtlist
EZZ2500I NETSTAT CS V2R10 TCPCS
PORT# PROT USER     FLAGS    RANGE       IP ADDRESS      SAF NAME
00600 TCP  *        AF                                   MYAPP
```

TN3270 Secure Telnet Port Access

Similar to PORTACCESS, z/OS CS ensures the user attempting to connect to a secure port is allowed access to the port. This support is used in conjunction with the TN3270 SSL client authentication support and prevents a client from seeing and getting past the USSMSG without the required authorization. If a SAF resource for the port exists in the SERVAUTH class, any user attempting to use the specified secure port number requires READ access to the resource. The SERVAUTH resource name has the following format:

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

where *nnnnn* is the port number with leading zeros.

For example, the security product profile name for port 23 running on the TCP stack named TCPCS on system MVSVIC97 is:

```
EZB.TN3270.MVSVIC97.TCPCS.PORT00023
```

Wildcarding can be used in the resource name (for example, EZB.TN3270.MVSVIC97.TCPCS.PORT*) if an installation's security product supports wildcards in profile names. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Configuring the Site Host Table (HOSTS.LOCAL) (Optional)

Why Configure a HOSTS.LOCAL Data Set?

You can set up the local hosts file to support local host name resolution. If you use the local hosts file for this purpose, your socket applications will only be able to resolve names and IP addresses that appear in your local hosts file.

If you need to resolve host names outside your local area, you can configure the resolver to use a domain name server (see the NSINTERADDR statement). If you use a domain name server, you do not need to set up any host definitions in your resolver configuration, but you may still do so.

If you have configured your resolver to use a name server, it will always try to do so, unless your applications were written with a RESOLVE_VIA_LOOKUP symbol in the source code. If this is the case, all name resolution calls from such a program will always use the local hosts file. This is probably not a technique you will see for standard socket applications, but it may be a technique you could find useful for when you develop your own socket applications for the z/OS UNIX environment.

It might be a good idea to have a local hosts file available for the resolver to use if the name server is not reachable. If the name server does not respond to name resolution requests, the resolver tries to use the local hosts file. If the name server is reachable but returns a negative reply for a name resolution request, the resolver tries to resolve the unqualified name via the local hosts file, if such a file is present.

Assume you try to resolve the host name *friendly* and your DOMAINORIGIN is *my.wood.com*, the resolver sends a query to the name server for *friendly.my.wood.com*. If the name server returns a negative reply (the name is not registered), the resolver looks into the local hosts file for an entry of *friendly* and, if not found, for an entry of *friendly.my.wood.com*.

Due to the flexibility of the Domain Name System, it is recommended you use a domain name server. If you set up a small TCP/IP network, the simplicity of the local hosts file approach is preferable.

Creating the Site Host Table (HOSTS.LOCAL)

The site table is generated from the *hlq*.HOSTS.LOCAL data set. This data set contains descriptions of local host entries in the HOSTS format. A sample HOSTS.LOCAL data set is created during installation. The following sections describe how to update the sample *hlq*.HOSTS.LOCAL data set and use it to generate the two data sets, *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO, which function as your site table.

HOST Entries

One line of the *hlq*.HOSTS.LOCAL data set is used for each distinct host and ends with four colons. Each host can have multiple IP addresses and multiple names. The line for each host has three essential fields, separated by colons. These fields are:

- The keyword *HOST*
- A list, separated by commas, of IP addresses for that host
- A list, separated by commas, of fully qualified names for that host

For example, if you have two local hosts, LOCAL1 (IP addresses 192.6.77.4 and 192.8.4.1) and LOCAL2 (with an alias LOCALB and IP address 192.6.77.2), append the following lines to the *hlq*.HOSTS.LOCAL data set:

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

Note: The maximum length for a host allowed in the HOST tables is 24 characters. However, the name server does not have a maximum character length.

NET and GATEWAY Entries

The NET and GATEWAY statements are not used by TCP/IP for z/OS applications. However, some socket calls require the NET entries. If your programs do not need the NET and GATEWAY statements, delete them before invoking MAKESITE.

Sample HOSTS.LOCAL Data Set (HOSTS): Following is the sample HOSTS.LOCAL data set provided in SEZAINST(HOSTS):

```
; HOSTS.LOCAL
; -----------
; COPYRIGHT = NONE.
;
; The format of this file is documented in RFC 952, "DoD Internet
; Host Table Specification".
;
; The format for entries is:
;
; NET : ADDR : NETNAME :
; GATEWAY : ADDR, ALT-ADDR : HOSTNAME : CPUTYPE : OPSYS : PROTOCOLS :
; HOST : ADDR, ALT-ADDR : HOSTNAME, NICKNAME : CPUTYPE : OPSYS : PROTOCOLS :
;
; Where:
;   ADDR, ALT-ADDR = IP address in decimal, e.g., 26.0.0.73
;   HOSTNAME, NICKNAME = the fully qualified host name and any nicknames
;   CPUTYPE = machine type (PDP-11/70, VAX-11/780, IBM-3090, C/30, etc.)
;   OPSYS = operating system (UNIX, TOPS20, TENEX, VM/SP, etc.)
;   PROTOCOLS = transport/service (TCP/TELNET,TCP/FTP, etc.)
;   : (colon) = field delimiter
;   :: (2 colons) = null field
; *** CPUTYPE, OPSYS, and PROTOCOLS are optional fields.
;
;   MAKESITE does not allow continuation lines, as described in
;   note 2 of the section "GRAMMATICAL HOST TABLE SPECIFICATION"
;   in RFC 952.  Entries should be specified on a single line of
;   up to a maximum of 512 characters per line.
;
;
; Note:  The NET and GATEWAY statements are not used by the TCP/IP for
;        MVS applications.  However, some socket calls require the NET
;        entries.  For better performance, if your programs do not need
;        the NET and GATEWAY statements, delete them before running
;        the MAKESITE program.
;
```

```
;
HOST : 9.67.43.100 : NAMESERVER ::::
HOST : 9.67.43.126 : RALEIGH ::::
HOST : 129.34.128.245, 129.34.128.246 : YORKTOWN, WATSON ::::
;
NET  : 9.67.43.0 : RALEIGH.IBM.COM :
;
GATEWAY : 129.34.0.0 : YORKTOWN-GATEWAY ::::
;
```

Using MAKESITE

Because many servers and commands allocate *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO, it is important not to overwrite or delete these data sets while TCP/IP is running. To avoid disrupting any active users, use an *HLQ*=parm that is different than your active *hlq*. This allows you to swap names (by renaming the old HOSTS data sets and then renaming the new HOSTS data sets) without starting and stopping TCP/IP.

Use MAKESITE as a TSO command or in a batch job to generate new *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO data sets. The parameters are the same for either a TSO command or a batch job invocation of MAKESITE. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

After you make changes to your *hlq*.HOSTS.LOCAL data set, you must generate and install new *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO data sets.

Search Order for HOSTS.SITEINFO

The z/OS UNIX search path for HOSTS.SITEINFO configuration is:

1. Value of the environment variable X_SITE (if set).

   This environment variable can contain the information that, in a non-z/OS UNIX environment, was contained in a HOSTS.SITEINFO file that was created by MAKESITE.
2. The HFS file /etc/hosts (if it exists).

   This is the equivalent to the z/OS UNIX file that can be used in place of a HOSTS.SITEINFO file.
3. *userid*.HOSTS.SITEINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
4. *tcpip*.HOSTS.SITEINFO.

   tcpip represents the value of the DATASETPREFIX statement in a TCPIP.DATA file. The default is TCPIP.

HOSTS.SITEINFO information is used by the following functions:

- gethostbyname()
- sethostent()
- gethostent()
- endhostent()
- getnetbyname()

Search Order for HOSTS.ADDRINFO

The z/OS UNIX search path for HOSTS.ADDRINFO configuration is:

1. Value of the environment variable X_ADDR (if set).

   This environment variable can contain the information that, in a non-z/OS UNIX environment, was contained in a HOSTS.ADDRINFO file that was created by MAKESITE.

2. The HFS file /etc/hosts (if it exists).

   This is the equivalent to the z/OS UNIX file that can be used in place of a HOSTS.ADDRINFO file.
3. *userid*.HOSTS.ADDRINFO, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
4. *tcpip*.HOSTS.ADDRINFO

   tcpip represents the value of the DATASETPREFIX statement in a TCPIP.DATA file. The default is TCPIP.

HOSTS.ADDRINFO information is used by the following functions:

- getnetbyaddr()
- setnetent()
- getnetent()
- endnetent()
- gethostbyaddr()

Note that if /etc/hosts exists, it overrides both a HOSTS.SITEINFO data set and a HOSTS.ADDRINFO data set because it contains both types of configuration.

Verifying Your Configuration

At this point, your configuration files have been updated.

To verify a configuration, start the TCP/IP address space and ensure that you see the following message:

```
EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE
```

If the message is not displayed, the messages issued by the TCP/IP address space should describe why TCP/IP did not start.

Verify TCPIP.DATA and TCPIPJOBNAME

Note: For all of the following examples, the unchanged SAMPPROF shipped with TCP/IP is used as the PROFILE.TCPIP.

From the TSO ready prompt, verify that the TCPIP.DATA file specifies the correct TCP/IP address space by typing a NETSTAT command. If the wrong TCPIPJOBNAME is specified, you will see the following message:

```
netstat
Unable to open UDP socket to TCPCS2: TCPCS2 is not active.
READY
```

With a TCPIP.DATA file correctly specifying TCPIP, the following results are displayed. To ensure that the correct TCPIP.DATA file is found in the example, the SYSTCPDA is explicitly allocated.

```
alloc f(systcpda) dsn('sys1.tcpparms(tcpdata)') shr reuse
 READY
netstat home
 MVS TCP/IP NETSTAT CS V2R10       TCPIP NAME: TCPIP        17:10:57
 Home address list:
 Address          Link             Flg
 -------          ----             ---
 127.0.0.1        LOOPBACK         P
 READY
```

Verify /etc/resolv.conf

Next, verify the UNIX System Services environment with the onetstat commands. The following example shows the incorrect address space.

The /etc/resolv.conf file is shown.

Note: You only need to create /etc/resolv.conf if you want to maintain separate resolver configuration files for the different APIs.

```
EDIT       /etc/resolv.conf                                Columns 00001 00072
Command ===>                                                  Scroll ===> PAGE
****** ***************************** Top of Data ******************************
000001 TCPIPJOBNAME TCPCS2

onetstat home
Unable to open UDP socket to TCPCS2   : TCPCS2    is not active.
```

With the /etc/resolv.conf correctly specified with TCPIPJOBNAME TCPIP, the following is displayed:

```
onetstat
MVS TCP/IP onetstat CS V2R10       TCPIP Name:               13:15:51
User Id  Conn     Local Socket           Foreign Socket         State
-------  ----     ------------           --------------         -----
BPXOINIT 00000011 0.0.0.0..10007         0.0.0.0..0             Listen
TCPIP    0000000B 0.0.0.0..1025          0.0.0.0..0             Listen
TCPIP    00000010 0.0.0.0..23            0.0.0.0..0             Listen
TCPIP    0000000F 127.0.0.1..1025        127.0.0.1..1026        Establsh
TCPIP    0000000E 127.0.0.1..1026        127.0.0.1..1025        Establsh
Syslogd1 00000012 0.0.0.0..514           *..*                   UDP
```

Verifying PROFILE.TCPIP with netstat or onetstat

Many configuration values specified within the PROFILE.TCPIP file can be verified with the netstat command. To verify the physical network and hardware definitions, use the NETSTAT DEV command. To see operating characteristics use NETSTAT CONFIG. A version of Netstat runs in the TSO and UNIX environments and from the MVS operator environment. Refer to *z/OS Communications Server: IP Configuration Reference* for information about the syntax and output of the commands. Following is output from the TSO NETSTAT command. Use the netstat command in the environment with which you are most comfortable.

```
NETSTAT DEVLINKS
 MVS TCP/IP NETSTAT CS V2R10       TCPIP NAME: TCPIP        17:10:57
  DevName: LOOPBACK          DevType: LOOPBACK  DevNum: 0000
    DevStatus: Ready
    LnkName: LOOPBACK          LnkType: LOOPBACK    LnkStatus: Ready
      NetNum: 0   QueSize: 0   ByteIn: 0000014908   ByteOut: 0000014908
    BSD Routing Parameters:
      MTU Size: 00000             Metric: 00
      DestAddr: 0.0.0.0           SubnetMask: 0.0.0.0
    Multicast Specific:
      Multicast Capability: No

 READY
```

The SAMPROF provided defines only the LOOPBACK address (as shown in this example).

Your installation should have a DEVICE for each interface used by TCP/IP. Counters, BSD Routing Parameters, and Multicast information is given but will not be discussed here. See “Chapter 4. Routing” on page 143 and *z/OS Communications Server: IP Configuration Reference* for more information on these topics.

```
NETSTAT CONFIG
 MVS TCP/IP NETSTAT CS V2R10       TCPIP NAME: TCPCS        17:10:57
  TCP Configuration Table:
  DefaultRcvBufSize:  00016384  DefaultSndBufSize: 00016384
  DefltMaxRcvBufSize: 00262144
  MaxReTransmitTime:  120.000   MinReTransmitTime: 0.500
  RoundTripGain:      0.125     VarianceGain:      0.250
  VarianceMultiplier: 2.000     MaxSegLifeTime:    60.000
  DefaultKeepALive:   0.120     LogProtoErr:       00
  TcpFlags:           10

  UDP Configuration Table:
  DefaultRcvBufSize: 00065535  DefaultSndBufSize: 00065535
  CheckSum:          00000001  LogProtoErr:       01
  UdpFlags:          00

  IP Configuration Table:
  Forwarding: Yes    TimeToLive: 00064  RsmTimeOut:  00060
  FireWall:   00000  ArpTimeout: 01200  MaxRsmSize:  65535
  IgRedirect: 00000  SysplxRout: 00001  DoubleNop:   00000
  StopClawEr: 00000  SourceVipa: 00000  VarSubnet:   00001
  MultiPath:  No     PathMtuDsc: 00000
  DynamicXCF: 00000

  SMF Parameters:
  InitType: 00  TermType: 00  ClientType: 00  TcpIpStats: 00

  Global Configuration Information:
  TcpIpStats: 00
```

The output from the NETSTAT CONFIG command should show many of the settings specified in PROFILE.TCPIP or implicitly taken from default values. Values set by the PROFILE.TCPIP operating characteristics can be verified at this point.

Verifying Interfaces via PING and TRACERTE

PING and TRACERTE can be used to verify adapters or interfaces attached to the z/OS host. Again, oping and otracert can be used in the z/OS UNIX environments with identical results. Since the example shipped with TCPIP has only the LOOPBACK address, for this section a 3172 LCS has been defined.

```
NETSTAT DEV
 MVS TCP/IP NETSTAT CS V2R10       TCPIP NAME: TCPIP        17:10:57
  DevName: LOOPBACK          DevType: LOOPBACK  DevNum: 0000
    DevStatus: Ready
    LnkName: LOOPBACK          LnkType: LOOPBACK    LnkStatus: Ready
      NetNum: 0   QueSize: 0   ByteIn: 0000000000   ByteOut: 0000000000
    BSD Routing Parameters:
      MTU Size: 00000             Metric: 00
      DestAddr: 0.0.0.0           SubnetMask: 0.0.0.0
    Multicast Specific:
      Multicast Capability: No
  DevName: LCS1              DevType: LCS       DevNum: 0120
    DevStatus: Ready
    LnkName: TR1               LnkType: TR          LnkStatus: Ready
      NetNum: 1   QueSize: 0   ByteIn: 0000000148   ByteOut: 0000000056
      ArpMacAddress: Non-Canonical   SrBridgingCapability: Yes
      BroadcastCapability: Yes       BroadcastType: All Rings
    BSD Routing Parameters:
      MTU Size: 00000             Metric: 00
```

```
      DestAddr: 0.0.0.0           SubnetMask: 255.255.255.128
    Multicast Specific:
      Multicast Capability: Yes
      Group             RefCnt
      -----             ------
      224.0.0.1         0000000001

 NETSTAT HOME
 MVS TCP/IP NETSTAT CS V2R10       TCPIP NAME: TCPCS        17:10:57
 Home address list:
 Address          Link             Flg
 -------          ----             ---
 9.67.113.63      TR1              P
 127.0.0.1        LOOPBACK

PING 9.67.113.63
 EZA0458I Ping CS V2R10: Pinging host 9.67.113.63.  Use ATTN to interrupt.
 EZA0463I PING: Ping #1 response took 0.001 seconds. Successes so far 1.
 READY
PING 127.0.0.1
 EZA0458I Ping CS V2R10: Pinging host 127.0.0.1.  Use ATTN to interrupt.
 EZA0463I PING: Ping #1 response took 0.001 seconds. Successes so far 1.
 READY
```

The TSO PING command uses the Pascal socket API so VMCF must be started for the command to be successful. If VMCF is not started, an ABEND.D6 will occur.

```
TRACERTE 9.67.113.63
 EZA0484I Trace route to 9.67.113.63 (9.67.113.63)
 EZA0505I 1   (9.67.113.63) 0 ms 0 ms 19 ms
 EZA0516I
 READY
TRACERTE 127.0.0.1
 EZA0484I Trace route to 127.0.0.1 (127.0.0.1)
 EZA0505I 1   (127.0.0.1) 3 ms 1 ms 12 ms
 EZA0516I
 READY
```

Given that your PROFILE.TCPIP file contains the interfaces of your installation and that the TCPIP.DATA file contains the correct TCPIPJOBNAME, the TCPIP address space is configured and you can go on to configuring routes, servers, and so on.

Verifying Local Name Resolution via TESTSITE

Use the TESTSITE command to verify that the *hlq*.HOSTS.ADDRINFO and *hlq*.HOSTS.SITEINFO data sets can correctly resolve the name of a host, gateway, or net. For more information on the TESTSITE command, refer to *z/OS Communications Server: IP Configuration Reference*.

Verifying PROFILE.TCPIP and TCPIP.DATA Using HOMETEST

Use the HOMETEST command to verify PROFILE.TCPIP and TCPIP.DATA.

Because HOMETEST is an MVS (and not a z/OS UNIX) application, it does not use RESOLVER_CONFIG or /etc/resolv.conf to do name resolution. In this example, it uses SYS1.TCPPARMS(TCPDATA) for its TCPIP.DATA.

```
hometest

Running IBM MVS TCP/IP CS V2R10 TCP/IP Configuration Tester

 The TCP/IP system parameter file used will be "SYS1.TCPPARMS(TCPDATA)".
 The FTP configuration parameter file used will be "SYS1.TCPPARMS(FTPDATA)".
```

```
TCP Host Name is: FRIENDLY.MY.WOOD.COM

Using Name Server to Resolve FRIENDLY.MY.WOOD.COM
The following IP addresses correspond to TCP Host Name: FRIENDLY.MY.WOOD.COM
9.67.113.63

The following IP addresses are the HOME IP addresses defined in PROFILE.TCPIP:
9.67.113.63
12.0.0.1
9.67.1.1
127.0.0.1

All IP addresses for FRIENDLY.MY.WOOD.COM are in the HOME list!

Hometest was successful - all Tests Passed!
```

Verifying Your X Windows System Installation (Optional)

Note: You cannot verify your X windows until after routing and DNS setup. Support is provided for two versions of the X Windows® System and the corresponding OSF/Motif. The current support, provided as part of the base IP support in z/OS CS, is for X Windows System Version 11 Release 6 and OSF/Motif Version 1.2. Support for X Windows System Version 11 Release 4 and OSF/Motif Version 1.1 is available as feature HTCP34X.

Verifying the X Windows X11R4 System Installation

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

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

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

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

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

Verifying the X Windows X11R6 System Installation

To verify the installation of the X Windows X11R6 System:

1. Ensure that a host (the workstation) with an X Windows System server that supports X11R6 is properly configured and reachable by the MVS system. From the workstation, use Telnet to access the MVS system, and open an z/OS UNIX shell on the MVS system.
2. From the z/OS UNIX shell, export the DISPLAY environment variable using either the network name or the qualified IP address of the workstation as shown in the following example:

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

   In this example, *royal.csc.ibm.com* is the name of the workstation running the X Windows server. The display is indicated by *:0.0*, and is specified this way in almost all cases.

3. Authorize the MVS system to access the workstation by executing the XHOST command, and specify either the name of the MVS system or a plus sign (+) as shown in the following example.

   ```
   xhost +
   ```

   Note: The *+* option turns off security for this workstation and allows any X client to display here.

4. The sample X Windows clients are shipped in the directory */usr/lpp/tcpip/X11R6/Xamples/demos*. Change into this directory. There are four sample program directories, xsamp1, xsamp2, xsamp3, and pexsamp. Change to the xsamp1 directory. Verify that there are files named Makefile and xsamp1.c, and then execute the following command:

   ```
   make
   ```

5. Execute the program using the following command:

   ```
   xsamp1
   ```

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

Chapter 3. Virtual IP Addressing

This chapter contains information about the following topics:

- Terminology
- Introduction to VIPA
- Moving VIPA (upon outage of TCP/IP)
- Static VIPAs, Dynamic VIPAs (DVIPAs), and Distributed Dynamic VIPAs
- Using static VIPAs
- Using Dynamic VIPAs (DVIPAs)
- Choosing Which Form of Dynamic VIPA to Use
- Configuring Distributed DVIPAs — Sysplex Distributor
- Resolution of DVIPA conflicts
- Other considerations
- DVIPAs and routing protocols

Terminology

Virtual IP Address (VIPA)
A VIPA is a generic term that refers to an internet address on an z/OS host that is not associated with a physical adapter. There are two types of VIPAs:

- A *Static VIPA* cannot be changed except through a VARY OBEY operator command.
- A *Dynamic VIPA (DVIPA)* can move to other TCP/IP stack members in a sysplex or it can be activated by an application program or by a supplied utility. Dynamic VIPAs are used to implement Sysplex Distributor as described in “Considerations for VIPA” on page 50.

Distributed DVIPA
A distributed DVIPA, which is a special type of DVIPA, can distribute connections within a Sysplex.

Dynamic routing
VIPAs are designed to interoperate with a dynamic routing daemon. Therefore, it is highly recommended that a routing daemon be used on an z/OS host that uses VIPAs.

Introduction to VIPA

Traditionally, an IP address is associated with each end of a physical link (or each point of access to a shared-medium LAN), and the IP addresses are unique across the entire visible network, which can be the Internet or a closed intranet. The majority of IP hosts have a single point of attachment to the network, but some hosts (particularly large server hosts) have more than one link into the network. Because the virtual device exists only in software, it is always active and never experiences a physical failure. A VIPA has no single physical network attachment associated with it. Also, the TCP/IP stack does not maintain interface counters for VIPA interfaces (VIRTUAL links). A TCP/IP host with multiple points of attachment also has multiple IP addresses, one for each link.

Within the IP routing network, failure of any intermediate link or adapter disrupts end user service only if there is not an alternate path through the routing network.

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

The Virtual IP Address (VIPA), introduced in TCP/IP for MVS V3R1, removes the adapter as a single point of failure by providing an IP address that is associated with a stack without associating it with a specific physical network attachment. Because the virtual device exists only in software, it is always active and never experiences a physical failure. A VIPA has no single physical network attachment associated with it.

To the routing network, a VIPA appears to be a host address indirectly attached to the z/OS. When a packet with a VIPA destination reaches the stack, the IP layer recognizes the address and passes it to the protocol layer in the stack.

The failure of the physical interface can be extended to the failure of the TCP/IP address space, the entire z/OS, or for planned outages. A VIPA just needs to move to a backup stack, and the routes to the VIPA need to be updated. Then clients can transparently connect to the backup stack. This process is known as VIPA Takeover.Support for VIPA Takeover was introduced as an APAR on TCP/IP for MVS V3R1.

VIPA Takeover was been improved with the introduction of Dynamic Virtual IP Address (DVIPA) in CS/390 V2R8 and Distributed Dynamic Virtual IP Address (Distributed DVIPA) in CS/390 V2R10. The DVIPA function improves VIPA Takeover by allowing a system programmer to plan for system outages and provide for backup systems to take over without operator intervention or external automation. The Distributed DVIPA function allows the connections for a single DVIPA to be serviced by applications on several stacks listed in the configuration statement (the distribution list). This adds the benefit of limiting the scope of an application or stack failure, while also providing enhanced work load balancing.

In general, z/OS configured with VIPA provides the following advantages:

- Automatic and transparent recovery from device and adapter failure.

  When a device (for example, 3172, or channel-attached 2216) or adapter (for example, a Token Ring or FDDI card) fails, if there is another device or link that provides the alternate paths to the destination:

  – IP will detect the failure, find an alternate path for each network, and route outbound traffic to hosts and routers on those networks via alternate paths.

  – Inbound and outbound traffic will not need to reestablish the active TCP connections that were using the failed device.

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

  Assuming that an alternate stack is installed to serve as a backup, the use of VIPAs enables the backup stack to activate the VIPA address of the failed stack.

  Connections on the failed primary stack will be disrupted but they can be reestablished on the backup using the same IP as the destination. In addition,

the temporarily reassigned VIPA address can be restored to the primary stack after the cause of failure has been removed.

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

- Limited scope of a stack or application failure.

  If a DVIPA is distributed among several stacks, the failure of only one stack affects only the subset of clients connected to that stack. If the distributing stack experiences the failure, a backup assumes control of the distribution and maintains all existing connections.
- Enhanced workload management through distribution of connection requests.

  With a single DVIPA being serviced by multiple stacks, connection requests and associated workloads can be spread across multiple z/OS images according to Workload Manager (WLM) and Service Level Agreement policies (for example, QOS).
- Allows the non-disruptive movement of an application server to another stack so that workload can be drained from a system in preparation for a planned outage.

Moving a VIPA (For TCP/IP Outage)

While a VIPA provides non-disruptive rerouting of IP data during failure of a physical interface, failure of the stack or the associated z/OS (including planned outages) will disrupt connections or UDP sessions to applications on the failing stack. While failure of the TCP connection or UDP session will be visible to the clients, the duration of the outage is determined by how long the client application is unable to reconnect to an appropriate server application. Because it is common in large enterprises to have multiple instances of an application residing on different z/OS images, if the VIPA address can be moved to another stack that supports the application, the clients can reconnect and the perceived outage will be over.

An IP address associated with a particular physical device is unavailable until the owning stack is restarted; however, a VIPA is not associated with any particular physical interface. If a failure of a stack is detected, and a suitable application already is active on another stack, the VIPA can be moved. Connections on the failed stack will be disrupted, but they can be reestablished on the backup stack using the original VIPA.

Movement of a static VIPA to a backup stack can be accomplished by using VARY OBEY commands on the backup. The OBEY file must contain an appropriate set of DEVICE, LINK, HOME, and BSDROUTING statements. If OMPROUTE is used as the routing daemon, an appropriate interface statement is needed in the OMPROUTE configuration file. If the TCP/IP configuration file with the statements defining the VIPA is created in advance, the transfer can be accomplished via automation. This procedure is documented in *z/OS Communications Server: IP Configuration Reference*. Movement of a DVIPA, on the other hand, can be accomplished by configuring a stack to backup a specific DVIPA that is defined on another stack. In this case, failure of the defining stack causes the DVIPA to move without operator intervention or extra automation. See “Planning for Dynamic VIPA Takeover” on page 115 for more information. Regardless of the type of VIPA to be moved, it is up to the system programmer or operator to ensure that the VIPA is moved to a backup stack that has the appropriate server applications.

In the absence of a failure, a VIPA is just like any other IP address, and routing for a VIPA is the same as for an IP address associated with a physical link.

Static VIPAs, Dynamic VIPAs (DVIPAs), Distributed DVIPAs

z/OS TCP/IP stack supports two types of VIPAs: static and dynamic. Dynamic VIPAs (DVIPAs) can be used to distribute connections in a sysplex. This is referred to as a Distributed DVIPA.

All three VIPAs can coexist on a given stack, but there are differences in how these VIPAs are configured and used.

Static VIPAs have the following characteristics:

- They can be activated during TCP/IP initialization or VARY OBEY processing and are configured using DEVICE, LINK, and HOME statements along with either BSDROUTINGPARMS statements or appropriate OMPROUTE configuration statements.
- Using the SOURCEVIPA configuration option, static VIPAs can be used as the source IP address for outbound datagrams for TCP, RAW, UDP (except routing protocols), and ICMP requests. This provides tolerance of device and adapter failures for requests of connections originating at an z/OS TCP/IP stack.
- They can be moved to a backup stack after the original owning stack has failed, by using VARY OBEY processing to configure the VIPA on the backup stack and updating the routers.
- The number of static VIPAs on a stack is limited only by the range of host IP addresses that are available for that host.

Dynamic VIPAs have the following characteristics:

- They can be configured to be moved dynamically from a failing stack to a backup stack within the same sysplex without operator intervention or external automation.
- They can be dynamically activated by an application program.
- They can distribute connections within a sysplex.
- Unlike static VIPAs, Dynamic VIPAs:
  - Cannot be used as a VIPA for SOURCEVIPA.
  - Are limited to 256 per stack.
  - A dynamic VIPA cannot be specified as the VIPA used by Enterprise Extender for connectivity purposes. (See “Configuring Static VIPAs for Enterprise Extender” on page 114 for details.)

Distributed DVIPAs, introduced with z/OS V2R10, have the following characteristics:

- Have all the characteristics of DVIPAs, but cannot be dynamically activated by an application program.
- One stack defines a DVIPA and advertises its existence to the network. Stacks in the target distribution list activate the DVIPA and accept connection requests.
- Connection workload can be spread across several stacks.

See “Configuring Distributed DVIPAs — Sysplex Distributor” on page 123 for more detailed descriptions.

Using Static VIPAs

The following sections describe how to configure static VIPAs, the special case of static VIPAs and Enterprise Extender, and how to implement static VIPA Takeover.

Because a VIPA is associated with an z/OS TCP/IP stack and it is not associated with a specific physical network attachment, it can be moved to a stack on any image in the sysplex or *even* to any z/OS TCP/IP stack if the address fits into the network configuration.

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

To configure a static VIPA address in one stack, follow these steps:

1. Add VIPA DEVICE, LINK, HOME, and BSDROUTINGPARMS statements for each static VIPA to be defined.

   Note: A VIPA link cannot be coded in the GATEWAY or BEGINROUTES statement.
2. If tolerance of device and adapter failures is desired for requests or connections originating at an z/OS TCP/IP stack, specify the SOURCEVIPA option in the IPCONFIG statement. For this option to work properly, the receiving nodes in the network must be configured to recognize the SOURCEVIPA addresses using the static or dynamic routing protocols. Otherwise, timeouts for the connection or request responses will occur as a result of the VIPA addresses being network unreachable. For more information on configuring SOURCEVIPA addresses, refer to *z/OS Communications Server: IP Configuration Reference*.
3. For host name resolution of a VIPA address, configure the domain name servers to associate the host name with the VIPA.
4. Configure the routing daemon to advertise the presence of the VIPA.

Figure 16 on page 114 illustrates a sample configuration showing multiple network attachments using a single static VIPA address. It will be used as the TCPCS system throughout this chapter.

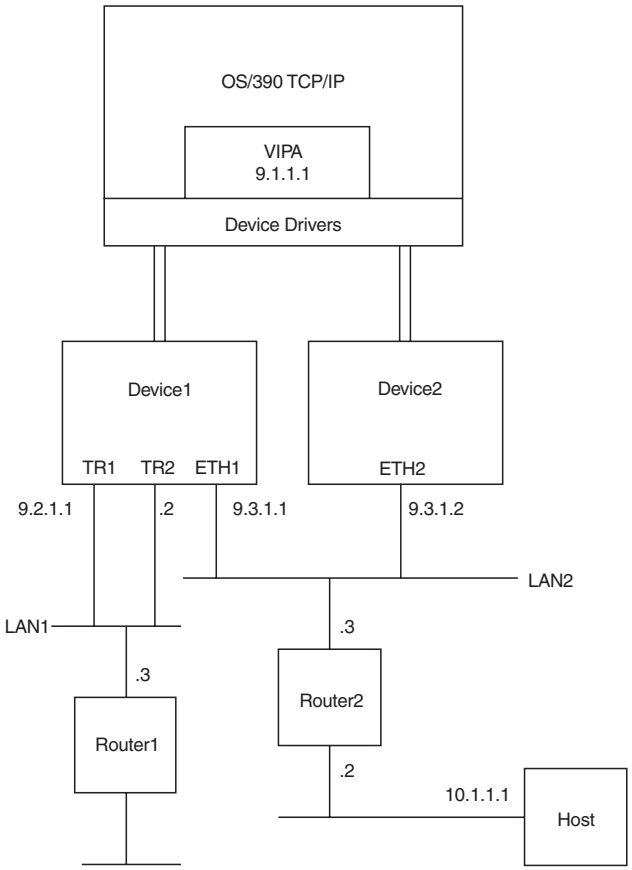


Figure 16. Static VIPA Configuration

Configuring Static VIPAs for Enterprise Extender

Defining at least one static VIPA is required by VTAM to access the IP network. Since VTAM does not move within a sysplex, a dynamic VIPA cannot be used. VTAM will use the VIPA address specified on the VTAM IPADDR start option. If the option is not used, VTAM will use the first static VIPA in the HOME list. If remote APPN® nodes use a host name and not a host address to define the destination of an Enterprise Extender connection, the domain name server must return the VIPA address used by VTAM for the host name.

For more information about Enterprise Extender, refer to the following:

- *z/OS Communications Server: SNA Network Implementation Guide*
- http://www.ibm.com/software/network/library/whitepapers/eextender.html
- http://www.ibm.com/software/network/library/whitepapers/eemsthtm/eemst.htm
- IBM Redbook, *SNA and TCP/IP Integration (SG24–5291–00)*

Note: This book is also available at http://www.ibm.com/redbooks.

Planning for Static VIPA Takeover and Takeback

Because a VIPA is associated with an z/OS TCP/IP stack and is not associated with a specific physical network attachment, it can be moved to a stack on any image in the sysplex or even to any z/OS TCP/IP stack as long as the address fits into the network configuration. Moving a static VIPA can be done manually by an operator

or by customer-programmed automation. Movement of the static VIPA allows other hosts that were connected to the primary stack to reestablish connections with a backup TCP/IP stack using the same VIPA. After the primary TCP/IP stack has been restored, the reassigned VIPA address can be moved back.

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

- All connections on the failing host will be disrupted.
- The client can use any ephemeral port number when reestablishing the connection to the backup server.
- Having a different port number for the backup and primary server is not recommended. For example, if the primary FTP used port 21 and the backup FTP used port 1021, when backing up and restoring an z/OS TCP/IP stack, the client would have to know whether to use port 21 or 1021.

Using Dynamic VIPAs (DVIPAs)

DVIPA support allows:

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

Dynamic VIPAs (DVIPAs) are IP addresses like all other IP addresses associated with a TCP/IP, and they appear as though they had been defined at the end of the HOME list. Because TCP/IP searches the HOME list for each incoming IP packet to determine whether the packet belongs to this stack, having too many IP addresses in the HOME list might adversely affect the performance of all incoming IP traffic.

Configuring Dynamic VIPA (DVIPA) Support

Unlike static VIPAs, DVIPAs are not configured using DEVICE, LINK, and HOME statements. The configuration statements for the DVIPA support are contained within the VIPADYNAMIC and ENDVIPADYNAMIC block and consist of the following:

- VIPADEFINE and VIPABACKUP statements used to configure DVIPAs to be dynamically moved from a failing TCP/IP to a backup TCP/IP
- VIPARANGE used to specify a range of IP addresses which may be dynamically activated as a VIPA by an application program
- VIPADELETE used to delete existing DVIPAs
- VIPADISTRIBUTE used to configure a DVIPA as a distributed DVIPA and designate a target stack.

The following sections discuss how these statements are used to provide the DVIPA support. For syntax details, see *z/OS Communications Server: IP Configuration Reference*.

When Dynamic VIPAs (DVIPAs) are used for VIPA Takeover together with DNS/WLM in a sysplex, code all of the DVIPAs in the sysplex under each host name in the DNS forward domain data file for the cluster zone. This will circumvent manual intervention in the DNS data files when a DVIPA is taken over or given back and will not cause any undesirable effects in DNS/WLM function.

Planning for Dynamic VIPA Takeover

Movement by network management automation or operator intervention is not always desirable. Operator intervention takes time and is subject to errors.

Automation requires proper detection of the failure and is also prone to error if the failure does not produce the exact console messages anticipated.

Dynamic VIPA Takeover function addresses this problem. It is important to understand that Dynamic VIPA Takeover does not introduce function that could not be accomplished by operator action or automation. It just removes the dependency on human detection of the error or customer programming for automation. Dynamic VIPA Takeover is completely accomplished by the TCP/IP stacks.

DVIPA Takeover is possible when a DVIPA is configured as active (via VIPADEFINE) on one stack and as backup (via VIPABACKUP) on another stack within the sysplex. When the stack on which the DVIPA is active terminates, then the backup stack will automatically activate the DVIPA and notify the routing daemon. For DVIPA Takeover to be useful, the applications that service the DVIPA addresses must be available on the backup stacks. In the absence of the application, the DVIPA will be active, but client connections to the application will still fail.

A determination of how the workload will be distributed among the backup stacks when the primary stack fails should be made. It is possible to designate a single stack as a backup and move all the workload to it, or the workload can be spread among several stacks. In the first case, only one DVIPA must be configured with a VIPADEFINE statement on the primary stack, and only one VIPABACKUP statement is required on the backup stack. The second option requires the definition of a VIPABACKUP statement for each stack that will assume responsibility for a subset of the primary's workload.

After determining the workload distribution, each of the secondary stacks will require a VIPABACKUP statement for the DVIPA it will be supporting.

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

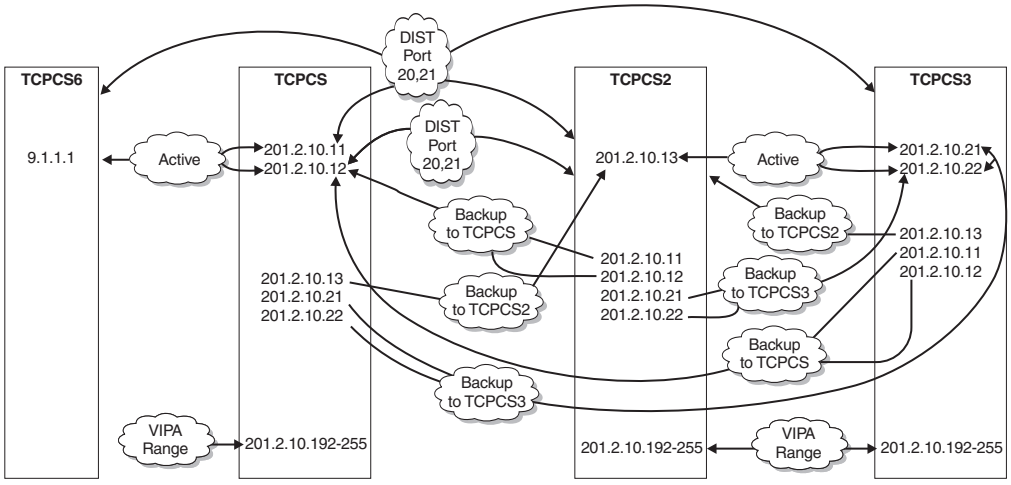


Figure 17. Sample DVIPA Addressing in a Sysplex Environment

Stack TCPCS:

```
Uses VIPADEFINE to define 201.2.10.11
Has a Web server running that binds to INADDR_ANY.
  Web client programs use 201.2.10.11 as their destination address.
Has an FTP server running that binds to INADDR_ANY.
  FTP client programs use 201.2.10.11 as their destination address.
```

Stack TCPCS3:

```
Uses VIPABACKUP to define 201.2.10.11 as backup for stack TCPCS.
Has a Web server running that binds to INADDR_ANY.
Has an FTP server running that binds to INADDR_ANY.
```

In the above scenario, when stack TCPCS goes down, stack TCPCS3 receives all new connection requests for both the Web and FTP servers. FTP and Web client programs continue to use 201.2.10.11 as their destination address, but they now connect to stack TCPCS3.

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

Stack TCPCS:

```
Uses VIPADEFINE to define 201.2.10.11 and 201.2.10.12
Has a Web server running that binds to INADDR_ANY.
  Web client programs use 201.2.10.11 as their destination address.
Has an FTP server running that binds to INADDR_ANY.
  FTP client programs use 201.2.10.12 as their destination address.
```

Stack TCPCS2:

```
Uses VIPABACKUP to define 201.2.10.11 as backup for stack TCPCS.
Has a Web server running that binds to INADDR_ANY.
```

Stack TCPCS3:

```
Uses VIPABACKUP to define 201.2.10.12 as backup for stack TCPCS.
Has an FTP server running that binds to INADDR_ANY.
```

In the above scenario, when stack TCPCS goes down, new connections for the Web server at 201.2.10.11 will connect with stack TCPCS2, and new connections for the FTP server at 201.2.10.12 will connect with stack TCPCS3.

Different Application Uses of IP Addresses and DVIPAs

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

Some applications will accept client requests on any IP address by binding to INADDR_ANY (for example, TN3270 or Web servers). The distinguishing feature of such an application is the function it provides (the particular set of SNA applications for TN3270 or the particular web pages for a Web server). If the function is replicated across multiple z/OS images in the sysplex, as is often the case for distributed workload, the DVIPA must merely be moved to a stack supporting the application. This scenario is called the Multiple Application-Instance Scenario. For the Multiple Application-Instance Scenario, the stacks in the sysplex do all the work of activating a DVIPA in the event of a failure.

For other types of applications, each application instance must have a unique IP address, either because it cannot bind to INADDR.ANY or because clients might establish a relationship to that server application instance that can span multiple TCP connections and the client must get connected back to the same server application instance while the relationship lasts. This scenario is called the Unique Application-Instance Scenario and uses DVIPAs that are activated with an ioctl or a bind().

To maintain the relationship between an application instance and its DVIPA, the application must indicate to the stack that the DVIPA needs to be activated. This occurs in the following cases:

- When the application instance issues a bind() function call and specifies an IP address that is not active on the stack. The stack will activate the address as a DVIPA, provided it meets certain criteria. When the application ends, the DVIPA is deleted.
- Some applications cannot be configured to issue bind() for a specific IP address, but are Unique Application-Instance Scenario applications. For such applications, a utility is provided (MODDVIPA), which issues SIOCSVIPA ioctl() to activate or deactivate the DVIPA. This utility can be included in a JCL procedure or OMVS script to activate the DVIPA before initiating the application. As long as the same JCL package or script is used to restart the application instance on another node in the event of a failure, the same DVIPA will be activated on the new stack. Root or APF authorization are required to execute the utility. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Configuring Dynamic VIPAs

To allow continued and unchanged operation of static VIPAs in z/OS TCP/IP, DVIPAs are defined with configuration statements in the PROFILE.TCPIP data set. An overview of the relevant configuration statements is provided in the following sections, and also see “Verifying the DVIPAs in a Sysplex” on page 134 for a description of the configuration statements. For an example of the VIPADYNAMIC/ENDVIPADYNAMIC configuration statements and display commands for Dynamic VIPA, see *z/OS Communications Server: IP Configuration Reference*.

Configuring the Multiple Application-Instance Scenario

For the Multiple Application-Instance Scenario, each instance is assigned a unique DVIPA. The VIPADEFINE keyword of the VIPADYNAMIC configuration statement is used to create the DVIPA on the stack where the DVIPA is normally expected to be active. When the VIPADEFINE statement is processed in a TCP/IP profile, corresponding DEVICE, LINK, HOME, and BSDROUTINGPARMS statements are generated automatically. Routing daemons are automatically informed.

Additional configuration is required on other stacks in the sysplex to indicate which stack should take over the DVIPA in the event of a failure. The VIPADYNAMIC statement has a VIPABACKUP keyword for this purpose. A VIPABACKUP configures the DVIPA but does not activate it until it is necessary. Because more than one TCP/IP can backup a single DVIPA, a rank parameter on the VIPABACKUP statement determines the order in which several stacks will assume responsibility for a DVIPA.

The stacks in the sysplex exchange information on all VIPADEFINEs and VIPABACKUPs defined in the sysplex, so that all are aware of which stack should

take over a particular DVIPA. The list of backup stacks for a specific DVIPA can be different from the list of backup stacks for all other DVIPAs.

In the Multiple Application-Instance Scenario, instances of the application in question are activated among sysplex nodes according to some plan, presumably related to balancing workload across available capacity. This activation is done independently of VIPA Takeover. Configure the associated DVIPAs as follows:

1. For each instance of a particular application to be supported via DVIPA, add a VIPADEFINE statement to the TCP/IP profile for the TCP/IP associated with the application instance.
2. For each of the Dynamic VIPAs in Step 1, determine which application instance or instances should take over the workload (considering probable capacity and any other application-related considerations). If more than one TCP/IP is to provide backup for a DVIPA, determine the order in which the selected TCP/IPs should be designated as backup. Add a VIPABACKUP statement to each TCP/IP that is to provide backup for the DVIPA, with appropriate rank values to determine the order. Do this for each of the DVIPAs in Step 1.
3. Perform steps 1 and 2 for each other application to be supported by DVIPAs.

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

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

Configuring the Unique Application-Instance Scenario

The Unique Application-Instance Scenario ties a DVIPA to a specific instance of an application. To isolate errors in configuring applications, TCP/IP needs a mechanism to identify permissible DVIPAs. This is provided with one or more VIPARANGE statements. The VIPARANGE statement identifies a range of IP addresses which can be activated as DVIPAs by an application instance. The VIPARANGE statement consists of a subnet mask and an IP address and thus defines a subnet for DVIPAs. More than one VIPARANGE statement with different ranges can be defined on a TCP/IP. VIPARANGE does not itself cause any DVIPAs to be activated. Rather, DVIPAs are activated either by an application issuing a bind() for a specific IP address, by use of the SIOCSVIPA ioctl() command issued by an authorized application, or by the MODDVIPA utility.

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

When an authorized application issues the SIOCSVIPA ioctl() command to create a DVIPA, or when the MODDVIPA utility is executed in JCL or an OMVS script to activate a DVIPA on behalf of an application instance, the current VIPARANGES are checked to see whether the IP address falls within one of them. If an appropriate VIPARANGE is found, and the IP address is not currently active on this TCP/IP or elsewhere in the sysplex as an IP address or a VIPADEFINE/VIPABACKUP Dynamic VIPA, then the IP address is activated as a DVIPA. However if no appropriate VIPARANGE is found on this TCP/IP, or if the IP address is currently defined on this TCP/IP or configured elsewhere in the sysplex as an IP address or a VIPADEFINE/VIPABACKUP Dynamic VIPA, then the request fails with errno and errnojr set to indicate the reason for the failure and the utility ends with a nonzero condition code. See "Dynamic VIPA Creation Results" on page 129 for more information.

Note: If the requested IP address has been activated as a Dynamic VIPA by a bind() or SIOCSVIPA ioctl elsewhere in the sysplex, the result depends on how the stacks were configured. See "Dynamic VIPA Creation Results" on page 129 for more information.

In the Unique Application-Instance Scenario, each application instance is assigned a unique IP address as its DVIPA. Before defining individual DVIPAs, one or more blocks of IP addresses must be defined for these DVIPAs, and the individual DVIPAs must be defined from within the blocks. Each block should be represented as a subnet, so that a VIPARANGE statement can be defined for it.

Follow these steps when setting up any unique application instances:

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

   Note: The dynamic VIPA must be within the VIPARANGE subnet. The broadcast address and the net prefix cannot be used.
3. Perform steps 1 and 2 until all application instances have been allocated a unique DVIPA.

The application restart strategy should ensure that the worst-case failure scenario does not attempt to activate more than 256 DVIPAs on a single stack. If such an attempt is made, activation of the 257th DVIPA will fail, with possible resulting loss of connectivity from clients to the server application.

Note: The limit of 256 DVIPAs on a single TCP/IP applies to all DVIPAs, whether defined by VIPADEFINE/ VIPABACKUP configuration statements, through a VIPADISTRIBUTE statement on another stack, by a bind() call, or by executing the MODDVIPA utility.

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

Using the 'SIOCSVIPA' Ioctl Command

An ioctl command 'SIOCSVIPA' allows an application to create or delete a Dynamic VIPA on the stack where the application is running. The application issuing the 'SIOCSVIPA' ioctl command must be authorized or running under a user with root authority. Refer to the *z/OS C/C++ Run-Time Library Reference* for more information.

To create a new Dynamic VIPA, the requested IP address must be within a subnet that has been previously specified by a VIPARANGE configuration statement in the PROFILE.TCPIP data set for this stack. The 'SIOCSVIPA' ioctl command can be used to delete any existing Dynamic VIPA on the stack, except for distributed DVIPAs.

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

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

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

Using the MODDVIPA Utility

You can use the MODDVIPA utility to activate or delete a Dynamic VIPA. The utility can be initiated from JCL or an OMVS script. MODDVIPA must be loaded from an authorized library and be invoked from a user ID with root authority.

Note: In CS/390 V2R8, this utility was called EZBXFDVP. The EZBXFDVP name will continue to work as it did in V2R8, but MODDVIPA is the preferred name and will be used throughout this book.

Input Parameters: The input parameters for the utility are:

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

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

Notes:

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

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

0 Success: The DVIPA was activated.

4 Warning: The requested DVIPA was not activated because the specified IP address is already active on this stack.

8 Error: The IP address was not defined as a DVIPA on this TCP/IP.

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

0 Success: The Dynamic VIPA was deleted.

8 Error: The requested DVIPA was not deleted.

Notes:

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

Examples

Within JCL:

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

From OMVS shell:

```
moddvipa -p TCPCS3 -c 1.2.3.4
```

Choosing Which Form of Dynamic VIPA Support to Use

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

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

- One or more applications bind to INADDR_ANY and exist on multiple TCP/IPs.
- Dynamic VIPA Takeover is desired.
- The DVIPA does not need to be deleted when the application is stopped.

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

- The application cannot bind to INADDR_ANY or Dynamic VIPA Takeover is not desired.
- The IP address to which the application binds can be controlled by the user.
- Automatic deletion of the Dynamic VIPA when the application is stopped is acceptable.

- A specific Dynamic VIPA address must be associated with a specific application.
- The application is not APF authorized, or not run under a user ID with Root authority.

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

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

Configuring Distributed DVIPAs — Sysplex Distributor

With the introduction of V2R10, a distributing function was added to Dynamic VIPAs. A Distributed DVIPA exists on several stacks, but is advertised outside the sysplex by only one stack. This stack receives all incoming connection requests and routes them to all the stacks in the distribution list for processing. This provides the benefit of distributing the workload of incoming requests and providing additional fail-safe precautions in the event of a server failure.

You can distribute connections destined for a Dynamic VIPA (DVIPA) by adding a VIPADISTribute configuration statement for a previously defined Dynamic VIPA. The order of the statements is important. The VIPA is first VIPADEFined and then VIPADISTributed. Another TCP/IP can act as a backup for the Distributed DVIPA by properly coding a VIPABackup statement; the backup will perform the routing function in the event of a failure. The options specified on a VIPADISTribute statement are inherited by a backup stack unless the second stack has its own VIPADISTribute statement, in which case it will use that VIPADISTribute statement for distributing. You can also code a VIPADISTribute statement with just the VIPABackup statement and not for the VIPADEFine statement. This would allow workload distribution only during a primary outage.

Following is an example of a properly coded Distributed VIPA:

```
IPCONFIG SYSPLEXROUTING DATAGRAMFWD DYNAMICXCF 193.9.200.4 255.255.255.240 1
VIPADYNAMIC
  VIPADEFINE  255.255.255.192 9.67.240.02
  VIPADISTRIBUTE DEFINE  9.67.240.02 PORT 20 21 8000 9000 DESTIP
          193.9.200.2
          193.9.200.4
          193.9.200.6
ENDVIPADYNAMIC
```

To enable the TCP/IP to forward connections, Datagram Forwarding must be enabled (specify DATAGRAMFWD in the IPCONFIG statement). To enable stacks to report their WLM and/or QOS information to the distributing stack specify SYSPLEXROUTING in the IPCONFIG statement in all the participating stacks (distributing and target). There are several configuration changes that can be made to affect the method the distributing stack will use to forward connections to the target stacks. In each of the following items, *all participating stacks* is used to refer to the distributing stack and all target stacks.

WLM-based Forwarding

To enable the distributing stack to forward connections based upon the workload of each of the target stacks, configure all participating stacks for WLM GOAL mode and specify SYSPLEXROUTING in the IPCONFIG statement in all participating stacks. This will register all participating stacks with WLM and will allow the distributing stack to request workload information from WLM.

WLM/QoS-based Forwarding

To enable the distributing stack to forward connections based upon a combination of workload information and network performance information (TCP retransmissions and timeouts), configure all participating stacks for WLM GOAL mode, specify SYSPLEXROUTING in the IPCONFIG statement in all participating stacks and also define a Sysplex Distributor Performance Policy on the target stacks. For information on configuring these policies, see "Defining Sysplex Distributor Policies Using PAGENT" on page 422.

Random Forwarding

In the absence of any of the above configuration changes, the distributing stack will randomly select one of the target stacks for each connection. Whether the distributing stack is performing WLM-based forwarding, WLM/QoS-based forwarding, or random forwarding, Sysplex Distributor Routing Policies can further affect the distribution of connections. Sysplex Distributor Routing Policies, configured on the distributing stack, are used to specify a set of target stacks for a given set of traffic. For example, all traffic destined to a given port/DVIPA from a specified subnet can be assigned one group of target stacks, while traffic for the same port/DVIPA from another subnet can be assigned to a different group of target stacks. For more information on configuring these types of policies, see "Defining Sysplex Distributor Policies Using PAGENT" on page 422.

The TCP/IP must also have a DYNAMICXCF address. When using Sysplex Distributor, do not define an IUTSAMEH link. These links will be created automatically from the DYNAMICXCF statement. This address is used by other distributing stacks as a destination point. Refer to *z/OS Communications Server: IP Configuration Reference* for directions for coding DYNAMICXCF on the IPCONFIG statement.

The VIPADISTribute statement specifies how new connection requests are routed to a set of candidate target stacks. The VIPADISTRIBUTEd DVIPA is followed by up to four ports, in this case the well-known ports for FTP and the ports for a custom application. Up to 32 target TCP/IPs follow the DESTIP keyword and are identified by their respective Dynamic XCF IP addresses. The VIPADISTribute statement may also specify DESTIP ALL, in which case all current and future stacks with activated Dynamic XCF may participate in the distribution as candidate target stacks. As an application listens to one of the specified ports on each listed TCP/IP, the routing TCP/IP begins to forward connections to that stack.

Note: In this example, both ports 20 and 21 are distributed for FTP. It is very common for applications to have both a communication and a data transfer port, and it is a common mistake for system programmers to forget about these ports. These applications will not work unless both ports are distributed. In the case of FTP, if only port 21 is distributed, clients would be able to connect to the server but they will not be able to transfer data. The connection protocol runs across port 21, but the data is transferred across port 20.

When this option is omitted, the routing stack randomly selects one of the listed destination stacks to service the new request. When some targets are running WLM compact mode and some are running WLM goal mode, the target stacks running WLM compact mode will not be selected to service any requests. Only WLM goal mode targets will be selected when both compact and goals modes exist.

If an application connects outbound from a target stack and uses a VIPADISTributed DVIPA as the source IP address, the VIPADISTRIBUTE statement on the distributing stack must contain the target stacks source port. Otherwise, the distributing stack will not be able to forward subsequent packets to the target for this connection. The FTP command is an example of this situation. FTP uses port 21 for control connections and port 20 for data connections. A control connection is established when a client connects to a server on a target stack (which the distributing stack has selected). When data needs to be transferred, a request is sent over the control connection to the server (on the target stack). The server then connects to the client using the DVIPA as its source address and port 20 as its source port. In order for subsequent packets to be forwarded across the data connection to the server, port 20 must be specified in the VIPADISTRIBUTE statement. The following example illustrates a VIPADISTRIBUTE statement for FTP and connections destined for DVIPA address 9.43.242.1:

```
VIPADEFINE  255.255.255.240 9.43.242.1
  VIPADISTRIBUTE DEFINE 9.43.242.1 PORT 20 21
   DESTIP 9.67.240.4
```

Resolution of Dynamic VIPA Conflicts

The same Dynamic VIPA can exist on more than one stack in the Sysplex, playing different roles on the different stacks. The TCP/IP stacks collaborate to prevent conflicting definitions. For example, at any given time only one stack will advertise a given Dynamic VIPA to the routers.

Potentially conflicting Dynamic VIPA definitions can arise during profile processing or as the result of changes within the sysplex due to a stack or application failure or as the result of movement of workload to a different stack. The following scenarios are examples of dynamic VIPA conflict resolution handled automatically by the TCP/IP stacks. For a summary of dynamic VIPA conflict identification and resolution, see “Dynamic VIPA Creation Results” on page 129.

Restart of the Original VIPADEFINE TCP/IP After an Outage

When a Dynamic VIPA is defined via VIPADEFINE on one TCP/IP, and other stacks are designated as backup via VIPABACKUP statements for the same dynamic VIPA, the stack with the highest backup rank for that DVIPA will activate it if or when the VIPADEFINE stack fails.

If the failed stack is later restarted with the same VIPADEFINE profile statement, it is likely that connections to that DVIPA will exist on the backup stack that now has the DVIPA activated and advertised to the routers. How and when ownership of the DVIPA is returned to the restarted stack is determined by how the DVIPA was originally configured.

VIPADEFINE MOVEABLE IMMEDIATE

If the DVIPA was originally configured with MOVEABLE IMMEDIATE, the following occurs:

- The DVIPA ownership is immediately transferred to the restarting stack which adds the DVIPA to its HOME list and the routers are dynamically notified. The

restarted stack receives all new connections for that DVIPA. The stack also can receive packets for existing connections, and it routes these to the backup stack to preserve those connections.

- At the same time, the backup stack notifies the routers that it no longer is the owner of the DVIPA.
  - If there are no current connections to the DVIPA, it is removed from the HOME list on the backup stack and it reverts to backup status.
  - If there are any existing connections, the DVIPA remains in the HOME list of the backup stack and the DVIPA is put into *Moving* status until the last existing connection is terminated. At that time, the DVIPA is removed from the HOME list and reverts to backup status.
- IBM recommends this form of a planned DVIPA take back occur only during low periods of connection activity. This gives the attached routers time to update their routing tables and avoid connections being reset due to receiving an ICMP_HOST_UNREACH from the router.

Notes:

1. MOVEABLE IMMEDIATE is the default for V2R10 and later.
2. The behavior described for MOVEABLE IMMEDIATE applies only when both the backup and the restarted stack are running V2R10 or later. If either the restarted stack or the backup stack is running V2R8, the behavior is the same as described in "VIPADEFINE MOVEABLE WHENIDLE".

VIPADEFINE MOVEABLE WHENIDLE

If the DVIPA was originally configured with MOVEABLE WHENIDLE (or the restarted or backup stack is running V2R8), the following occurs:

- If it appears that there are no active connections to the DVIPA on the backup stack:
  - The DVIPA is removed from the HOME list on the backup stack and reverts to backup status.
  - The restarted stack assumes ownership of the DVIPA by adding it to its HOME list and notifying the routers.
- If there are existing connections to the DVIPA on the backup stack:
  - Ownership of the DVIPA remains with the backup stack. The DVIPA on the restarting stack is placed in backup status at the head of the backup list for the DVIPA.
  - The backup stack periodically checks to see if it has any active connections to the DVIPA. (TCP/IP does *not* refuse new connections during this time because this would be the same as an outage.)

    When or if it appears that there are no active connections for the DVIPA, the following occurs:

    - The DVIPA is removed from the HOME list on the backup stack and reverts to backup status.
    - The restarted stack assumes ownership of the DVIPA by adding it to its HOME list and notifying the routers.

    Notes:

    1. A small period of time exists between the check for connections and the movement of the dynamic VIPA to the restarted stack. If connections are made to the old host (the backup stack) in this interval, they will be broken.
    2. During the time that TCP/IP is periodically checking for connections, TCP/IP does *not* refuse new connections because this would be the same

as an outage. If moving the work back to the restarted stack is more important than maintaining uninterrupted service to all clients, then the system operator can use VARY OBEY to delete the dynamic VIPA on the backup stack with the VIPADELETE profile statement. This causes the restarted stack to immediately activate the DVIPA. (Optionally, the VARY OBEY file can contain a VIPABACKUP statement following the VIPADELETE statement. This will restore the stack as a backup stack.)

Movement of Unique Application-Instance (BIND)

A dynamic VIPA is created when any application binds to a nonexistent, specific IP address falling within a configured VIPARANGE on that stack.

In the case of a stack failure, the same application could be started on another stack and (assuming the new stack also has an appropriate VIPARANGE configured) when the application binds to the same IP address, the dynamic VIPA is created on the second stack. Future client connections to that IP address are routed to the second stack where the application is now running.

However, if the same (or a different) application is started on a second stack and attempts to create the same dynamic VIPA via a bind() while it exists on the first stack, the end result is determined by how the VIPARANGE was configured on the stack where the first bind() occurred.

VIPARANGE (DEFINE) MOVEABLE NONDISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE NONDISRUPTIVE, the following occurs:

- The DVIPA ownership is immediately transferred to the second stack which adds the DVIPA to its HOME list and dynamically notifies the routers. This stack will now receive all new connections for the DVIPA.
- At the same time, the first stack notifies the routers that it no longer is the owner of the DVIPA, and puts the DVIPA into *moving* status. The DVIPA remains in *moving* status (and in the first stacks HOME list) until the application closes the socket.
- Existing connections on the first stack are preserved. If the second stack receives packets intended for existing connections, it routes the packets to the first stack.

Notes:

1. NONDISRUPTIVE is the default for V2R10 and later.
2. The applications that create dynamic VIPAs via BIND() do *not* have to be authorized (so you might want to specify DISRUPTIVE).
3. Both stacks must be running V2R10 or later to get non-disruptive behavior. If either stack is running V2R8, the result will be as described in “VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE”.

VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE DISRUPTIVE (or if either stack is running V2R8), the following occurs:

- The bind() request for the application on the second stack will fail.
- The DVIPA on the first stack is not affected.

Note: If movement of the application from the first to the second stack is intended, the application must be ended on the first stack before it is started on the second stack.

Movement of a Unique APF-Authorized Application Instance (ioctl)

APF-Authorized applications have the ability to activate a Dynamic VIPA via the SIOCSVIPA ioctl() either within the application itself or by invoking the MODDVIPA utility. Because this is a controlled environment, it is assumed configuration errors are minimized or avoided and the usage is correct. Thus, even if the requested DVIPA is currently active on another TCP/IP stack via BIND() or ioctll(), the DVIPA will be immediately activated on this stack. What happens on the other stack is determined by how the VIPARANGE was configured on that stack.

VIPARANGE (DEFINE) MOVEABLE NONDISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE NONDISRUPTIVE, the following occurs:

- The DVIPA ownership is immediately transferred to the second stack which adds the DVIPA to its HOME list and dynamically notifies the routers.
- At the same time, the first stack notifies the routers that it no longer is the owner of the DVIPA, and puts the DVIPA into *moving* status. The DVIPA remains in *moving* status (and in the first stack's HOME list) until the DVIPA is deleted on that stack via the VIPADELETE profile statement or the SIOCSVIPA ioctl DELETE option.
- Existing connections on the first stack are preserved. If the second stack receives packets intended for existing connections, it will route the packets to the first stack.

Notes:

1. NONDISRUPTIVE is the default for V2R10 and later.
2. Both stacks must be running V2R10 or later to get non-disruptive behavior. If either stack is running V2R8, the result will be as described in "VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE".

VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE DISRUPTIVE (or if either stack is running V2R8), the following occurs:

- The ioctl request for the application on the second stack succeeds. The DVIPA is added to the HOME list on the second stack, and the routers are dynamically notified.
- The DVIPA on the first stack is deleted.

Note: Any existing connections to the DVIPA on the first stack are broken.

Same Dynamic VIPA as VIPADEFINE and BIND()/SIOCSVIPA Ioctl, or MODDVIPA UTILITY

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

If this duplicate DVIPA address conflict occurs on the same TCP/IP, the second attempt might fail. If an IP address is specified in a VIPADEFINE and that same IP address has already been activated on the TCP/IP by an application via BIND(), the SIOCSVIPA ioctl, or the MODDVIPA utility is used, the VIPADEFINE will be rejected during VARY OBEY profile processing. If an IP address is activated via VIPADEFINE, and the application does a BIND(), ioctl(), or the MODDVIPA utility is

used, the BIND() will succeed, but the ioctl() will fail with a nonzero errno and the MODDVIPA utility will set a nonzero condition to indicate that the IP address already exists.

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

Dynamic VIPA Creation Results

Table 10 summarizes the results of attempting to create a Dynamic VIPA when it (or the same IP address for HOME statement) already exists in the sysplex.

Table 10. Summary of Dynamic VIPA Creation Results

First Action	Second Action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
bind()	bind()	Second bind() succeeds, but no new VIPA is created.	If both stacks are running V2R10 or later, and the first BIND DVIPA was created with MOVEABLE NONDISRUPTIVE: • On stack 2, bind() succeeds • On stack 1, the BIND VIPA remains in the HOME list (unadvertised) and any existing connections are preserved • New connections to that IP address go to the application on stack 2. Otherwise, second bind fails.
bind()	ioctl()	ioctl() fails with warning condition code, but the application associated with the ioctl is still able to use the Dynamic VIPA.	ioctl() succeeds, bind is deleted (even if BIND DVIPA was created as MOVEABLE NONDISRUPTIVE)
bind()	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
bind()	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
bind()	HOME	See note.	See note.

Table 10. Summary of Dynamic VIPA Creation Results (continued)

First Action	Second Action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
ioctl()	bind()	bind() succeeds, no new VIPA is created.	bind() fails.
ioctl()	ioctl()	Second ioctl() fails with warning condition code, but the application associated with the ioctl is still able to use the Dynamic VIPA.	Second ioctl() succeeds. If both stacks are running V2R10 or later, and the ioctl DVIPA on stack 1 was created with MOVEABLE NONDISRUPTIVE, the DVIPA on stack 1 remains in the HOME list (unadvertised) and any existing connections are preserved. Otherwise, the ioctl DVIPA on stack 1 is deleted and any existing connections are broken.
ioctl()	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
ioctl()	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
ioctl()	HOME	See note.	See note.
VIPADEFINE	bind()	bind() succeeds, but no new VIPA is created.	bind() fails.
VIPADEFINE	ioctl()	ioctl() fails.	ioctl() fails.
VIPADEFINE	VIPADEFINE	VIPADEFINE fails if the VIPADEFINE statement specifies a different mask or MOVEABLE setting than the first VIPADEFINE specified. If the second VIPADEFINE statement is an exact duplicate of the first, the second VIPADEFINE is ignored with no error message.	Second VIPADEFINE succeeds but activation on stack 2 might be deferred. If both stacks are running V2R10 or later, and the DVIPA was created on stack 1 as MOVEABLE IMMEDIATE: • Second VIPADEFINE is activated immediately • Any connections to the DVIPA on stack 1 are preserved. (DVIPA stays HOME list unadvertised) Otherwise, the second VIPADEFINE activation is deferred until there are no connections on stack 1, at which point, stack 1 reverts to backup status.
VIPADEFINE	VIPABACKUP	VIPABACKUP fails.	Both succeed.
VIPADEFINE	HOME	See note.	See note.

Table 10. Summary of Dynamic VIPA Creation Results (continued)

First Action	Second Action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
VIPABACKUP	bind()	bind() fails.	If the IP address is already active on the bind() stack, the bind() will succeed. Otherwise, the bind() fails.
VIPABACKUP	ioctl()	ioctl() fails.	ioctl() fails.
VIPABACKUP is backup status	VIPADEFINE	VIPADEFINE succeeds, replaces the VIPABACKUP.	VIPADEFINE succeeds.
VIPABACKUP in active status (after takeover)	VIPADEFINE	VIPADEFINE rejected	**Note:** The VIPABACKUP DVIPA is MOVEABLE IMMEDIATE or WHENIDLE depending how the original VIPADEFINE DVIPA was created. If both stacks are running V2R10 or later, and the VIPABACKUP DVIPA is MOVEABLE IMMEDIATE: • The VIPADEFINE is activated immediately • Any connections to the DVIPA on stack 1 are preserved (DVIPA stays in HOME list unadvertised) • When there are no more connections, stack 1 reverts to backup status Otherwise, the VIPADEFINE activation on stack 2 is deferred until there are no stack 1, at which point, stack 1 reverts to backup status.
VIPABACKUP	VIPABACKUP	Second VIPABACKUP succeeds.	Second VIPABACKUP succeeds.
VIPABACKUP	HOME	See note.	See note.
HOME	bind()	bind() succeeds, but no new VIPA is created.	bind() fails.
HOME	ioctl()	ioctl() fails.	ioctl() fails.
HOME	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
HOME	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
Note: Defining the same IP address in the HOME statement as an existing Dynamic VIPA will not be rejected by the TCP/IP stack, but it is likely to cause routing problems.			

Other Considerations

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

Mixture of Types of Dynamic VIPAs within Subnets

Any particular IP address can be used in only one way as a Dynamic VIPA. As described in previous sections, a Dynamic VIPA can be defined either via VIPADEFINE or by application action within a valid VIPARANGE, but not both. However, within a subnet defined as a VIPARANGE, some IP addresses can be used for VIPADEFINE, and others may be assigned to unique application instances, without conflict, as long as the limit of a total of 256 active and backup Dynamic VIPAs on a single TCP/IP is not exceeded. TCP/IP will make no attempt to reject a VIPADEFINE Dynamic VIPA that also falls within a VIPARANGE. This allows installations with limited availability of IP addresses to assign individual addresses to either application scenario, without having to define separate subnets and use up additional IP addresses in that manner.

MVS Failure and Sysplex Failure Management

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

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

For more information, refer to *z/OS Communications Server: IP Diagnosis*.

Applications and Dynamic VIPAs

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

UDP applications whose interactions consist of atomic interactions (a single request followed by one or more responses, with no state information maintained at the server between requests) can use Dynamic VIPAs in the Multiple Application-Instance Scenario. However, if the server application maintains state information between interactions (for example, NFS), then moving a Dynamic VIPA to another server might not work unless the client/server application protocol can

detect the discontinuity. In that case, the Unique Application-Instance Scenario might apply, which would require the restart of the server instance on another TCP/IP.

In addition, the following types of work are not appropriate for distribution with distributed dynamic VIPA:

- Applications that establish affinity with a particular TCP/IP stack, such as SNMP.
- Applications that bind to ephemeral ports.
- FTP servers that receive the PASV command. This command requests the FTP server to bind() on a data port that is not the default data port, or the one specified on the VIPADISTribute statement, and to wait for a connection rather than initiate one on receipt of a transfer command (for example, RETR).

Example of Configuring Dynamic and Distributed VIPAs

The TCP/IP profiles needed to implement Dynamic VIPA(DVIPA) on multiple systems in a sysplex are shown in the following examples. The VIPADEFINE and VIPABACKUP statements allow Automatic Dynamic VIPA Takeover to occur if needed (see “Configuring the Multiple Application-Instance Scenario” on page 118), and the VIPARANGE statements allow Dynamic VIPAs to be dynamically created by an application or by the MODDVIPA utility (see “Configuring the Unique Application-Instance Scenario” on page 119). The VIPADISTRIBUTE statements allow a single VIPA to be shared among several TCP/IPs.

```
TCPCS
 IPCONFIG  DATAGRAMFWD SYSPLEXROUTING
 DYNAMICXCF 193.9.200.1 255.255.255.0 14
 VIPADYNAMIC
  VIPADEFINE 255.255.255.240  201.2.10.11 201.2.10.12
  VIPADISTRIBUTE 201.2.10.11 PORT 20 21 DESTIP ALL
  VIPADISTRIBUTE 201.2.10.12 PORT 20 21 DESTIP 193.9.200.2
  VIPABACKUP 100 201.2.10.13
  VIPABACKUP 80  201.2.10.21
  VIPABACKUP 80  201.2.10.22
  VIPARANGE DEFINE 255.255.255.192  201.2.10.192
 ENDVIPADYNAMIC
TCPCS2
 IPCONFIG  DATAGRAMFWD SYSPLEXROUTING
 DYNAMICXCF 193.9.200.2 255.255.255.240 1
 VIPADYNAMIC
  VIPADEFINE 255.255.255.192  201.2.10.13
  VIPABACKUP 100 201.2.10.11  201.2.10.21
  VIPABACKUP 75  201.2.10.12  201.2.10.22
  VIPARANGE DEFINE 255.255.255.192  201.2.10.192
 ENDVIPADYNAMIC
TCPCS3
 IPCONFIG  DATAGRAMFWD SYSPLEXROUTING
 DYNAMICXCF 193.9.200.3 255.255.255.240 1
 VIPADYNAMIC
  VIPADEFINE 255.255.255.192   201.2.10.21  201.2.10.22
  VIPABACKUP  10  201.2.10.11  201.2.10.12  201.2.10.13
  VIPARANGE DEFINE 255.255.255.192          201.2.10.192
 ENDVIPADYNAMIC
TCPCS6
IPCONFIG SYSPLEXROUTING DATAGRAMFWD

DYNAMICXCF 193.9.200.6 255.255.255.224 1

TCPCS6 does not contain a VIPADYNAMIC definition
```

Start TCP/IP on each system as shown above.

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

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

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

Verifying the DVIPAs in a Sysplex

A display command parameter displays Dynamic VIPAs in the sysplex (see Figure 17 on page 116). In the following example taken from stack TCPCS, the ORIGIN lines show that 201.2.10.11 and 201.2.10.12 were created by VIPADEFINE on this stack, 201.2.10.193 was created by VIPARANGE ioctl (issued through the EZBXFDVP utility), and all of the others were created by VIPABACKUP statements. The command is:

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

The ORIGIN line indicates how the DVIPA is configured on the stack specified by `tcpname`. Each stack (TCPNAME) for each system (MVSNAME) is shown with its status (STATUS). Two other status values not shown in the following example are:

QUIESCING
: This DVIPA was a target for distribution and has been removed as a target. However, it is still servicing one or more connections for this DVIPA. The DVIPA will be removed when all connections complete.

MOVING
: This DVIPA was active on this stack and has been moved to another stack. Connections on this stack for this DVIPA prior to the move will still be serviced by this stack until completion.

The rank (RANK) indicates which of the stacks is eligible to take over if the stack on which the DVIPA is active stops. The stack with the highest rank is the one that will take over the DVIPA.

```
D TCPIP,TCPCS,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V2R10 874
VIPA DYNAMIC DISPLAY FROM TCPCS    AT MVS005
IPADDR: 201.2.10.11  LINKNAME: VIPLC9020A0B
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      BOTH
  TCPCS2   MVS004   BACKUP 100                                   DEST
```

```
  TCPCS3   MVS005   BACKUP 010                                    DEST
  TCPCS6   MVS006   ACTIVE                                       DEST
IPADDR: 201.2.10.12  LINKNAME: VIPLC9020A0C
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      DIST
  TCPCS2   MVS004   BACKUP 075                                   DEST
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.13
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS2   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 100
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.21
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS2   MVS004   BACKUP 100
  TCPCS    MVS004   BACKUP 080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 080
  TCPCS2   MVS004   BACKUP 075
IPADDR: 201.2.10.193  LINKNAME: VIPLC9020AC1
  ORIGIN: VIPARANGE ioctl
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.192 201.2.10.192
```

TCPCS2, TCPCS3, and TCPCS6 all display the same information about all the DVIPAs. ORIGIN fields are displayed for the DVIPAs that are configured on this stack.

```
D TCPIP,TCPCS2,SYS,VIPAD
EZZ8260I SYSPLEX CS V2R10 877
VIPA DYNAMIC DISPLAY FROM TCPCS2   AT MVS005
IPADDR: 201.2.10.11
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      BOTH
  TCPCS2   MVS004   BACKUP 100                                   DEST
  TCPCS3   MVS005   BACKUP 010                                   DEST
  TCPCS6   MVS006   ACTIVE                                       DEST
IPADDR: 201.2.10.12
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      DIST
  TCPCS2   MVS004   BACKUP 075                                   DEST
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.13  LINKNAME: VIPLC9020A0D
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS2   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 100
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.21
  ORIGIN: VIPABACKUP
```

```
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS2   MVS004   BACKUP 100
  TCPCS    MVS004   BACKUP 080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 080
  TCPCS2   MVS004   BACKUP 075
IPADDR: 201.2.10.193
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.192 201.2.10.192
```

In the following example, TCPCS6 knows about the DVIPAs on the other stacks. There are no DVIPAs configured on TCPCS6, thus, no ORIGIN fields displayed.

```
D TCPIP,TCPCS6,SYS,VIPAD
EZZ8260I SYSPLEX CS V2R10 880
VIPA DYNAMIC DISPLAY FROM TCPCS6   AT MVS005
IPADDR: 201.2.10.11
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      BOTH
  TCPCS2   MVS004   BACKUP 100                                  DEST
  TCPCS3   MVS005   BACKUP 010                                  DEST
  TCPCS6   MVS006   ACTIVE                                      DEST
IPADDR: 201.2.10.12
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      DIST
  TCPCS2   MVS004   BACKUP 075                                  DEST
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.13
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS2   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 100
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.21
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS2   MVS004   BACKUP 100
  TCPCS    MVS004   BACKUP 080
IPADDR: 201.2.10.22
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 080
  TCPCS2   MVS004   BACKUP 075
IPADDR: 201.2.10.193
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.192 201.2.10.192
```

Using NETSTAT Support to Verify Dynamic VIPA Configuration

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

Note: The VIPA configuration information has been broken out of the global configuration in order to simplify the display. Lowercase f indicates the overall configuration and the capital F indicates just the information for Dynamic VIPAs.

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

On stack TCP CS, using netstat on OMVS:

```
# netstat -p tcpcs -F
MVS TCP/IP onetstat CS V2R10       TCPIP Name: TCPCS            12:04:15
Dynamic VIPA Information:

VIPA Backup:
  IP Address       Rank
  ----------       ----
  201.2.10.13      000100
  201.2.10.21      000080
  201.2.10.22      000080

VIPA Define:
  IP Address       AddressMask      Moveable
  ----------       -----------      --------
  201.2.10.11      255.255.255.240  Immediate
  201.2.10.12      255.255.255.240  Immediate

VIPA Range:
  AddressMask      IP Address       Moveable
  -----------      ----------       --------
  255.255.255.192  201.2.10.192     NonDisr

VIPA Distribute:
  IP Address       Port   XCF Address
  ----------       ----   -----------
  201.2.10.11      00020  ALL
  201.2.10.11      00021  ALL
  201.2.10.12      00020  193.9.200.2
  201.2.10.12      00021  193.9.200.2
```

On stack, TCPCS2 from the console:

```
01.55.13  d tcpip,tcpcs2,net,vipadcfg
01.55.14  EZZ2500I NETSTAT CS V2R10 TCPCS2 764
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IP ADDRESS       RANK
    ----------       ----
    201.2.10.11      000100
    201.2.10.12      000075
    201.2.10.21      000100
    201.2.10.22      000075
  VIPA DEFINE:
    IP ADDRESS       ADDRESSMASK      MOVEABLE
    ----------       -----------      --------
    201.2.10.13      255.255.255.192  IMMEDIATE
  VIPA RANGE:
    ADDRESSMASK      IP ADDRESS       MOVEABLE
    -----------      ----------       --------
    255.255.255.192  201.2.10.192     NONDISR
```

On stack TCPCS3 from the console:

```
01.56.42  d tcpip,tcpcs3,net,vipadcfg
01.56.43  EZZ2500I NETSTAT CS V2R10 TCPCS3 767
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IP ADDRESS       RANK
    ----------       ----
    201.2.10.11      000010
    201.2.10.12      000010
    201.2.10.13      000010
  VIPA DEFINE:
    IP ADDRESS       ADDRESSMASK      MOVEABLE
    ----------       -----------      --------
    201.2.10.21      255.255.255.192  IMMEDIATE
    201.2.10.22      255.255.255.192  IMMEDIATE
  VIPA RANGE:
    ADDRESSMASK      IP ADDRESS       MOVEABLE
    -----------      ----------       --------
    255.255.255.192  201.2.10.192     NONDISR
```

On stack TCPCS6 from the console:

```
01.57.32  d tcpip,tcpcs6,net,vipadcfg
01.57.32  EZZ2500I NETSTAT CS V2R10 TCPCS6 770
```

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

On stack TCP CS using netstat on OMVS:

```
 # netstat -p tcpcs -v
MVS TCP/IP onetstat CS V2R10       TCPIP Name: TCPCS            02:03:07
IP Address      AddressMask     Status    Origination     DistStat
----------      -----------     ------    -----------     --------
201.2.10.11     255.255.255.240 Active    VIPADefine      Dist/Dest
201.2.10.12     255.255.255.240 Active    VIPADefine      Dist
201.2.10.13     255.255.255.192 Backup    VIPABackup
201.2.10.21     255.255.255.192 Backup    VIPABackup
201.2.10.22     255.255.255.192 Backup    VIPABackup
```

On stack TCPCS2 from the console:

```
02.04.09  d tcpip,tcpcs2,net,vipadyn
02.04.09  EZZ2500I NETSTAT CS V2R10 TCPCS2 795
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 BACKUP    VIPABACKUP      DEST
201.2.10.12     255.255.255.240 BACKUP    VIPABACKUP      DEST
201.2.10.13     255.255.255.192 ACTIVE    VIPADEFINE
```

On stack TCPCS, using the onetstat command:

```
# netstat -p TCPCS2 -v
MVS TCP/IP onetstat CS V2R10       TCPIP Name: TCPCS2           10:20:58
IP Address      AddressMask     Status    Origination     DistStat
----------      -----------     ------    -----------     --------
201.2.10.11     255.255.255.240 Backup    VIPABackup      Dest
201.2.10.12     255.255.255.240 Backup    VIPABackup      Dest
201.2.10.13     255.255.255.192 Active    VIPADefine
201.2.10.21     255.255.255.192 Backup    VIPABackup
201.2.10.22     255.255.255.192 Backup    VIPABackup
```

On stack TCPCS3 from the console:

```
02.05.21  d tcpip,tcpcs3,net,vipadyn
02.05.21  EZZ2500I NETSTAT CS V2R10 TCPCS3 798
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 BACKUP    VIPABACKUP      DEST
```

```
201.2.10.12     255.255.255.240 BACKUP    VIPABACKUP
201.2.10.13     255.255.255.192 BACKUP    VIPABACKUP
201.2.10.21     255.255.255.192 ACTIVE    VIPADEFINE
201.2.10.22     255.255.255.192 ACTIVE    VIPADEFINE
```

On stack TCPCS6 from the console:

```
02.05.58  d tcpip,tcpcs6,net,vipadyn
02.05.58  EZZ2500I NETSTAT CS V2R10 TCPCS6 801
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 ACTIVE                    DEST
```

Verifying Sysplex Distributor Workload

The netstat commands (TSO NETSTAT, z/OS UNIX onetstat, and MVS console D TCPIP,,Netstat) have a new VDPT (-O) report and a new VCRT (-V) report. Refer to *z/OS Communications Server: IP User's Guide* the for more information on these commands.

Run onetstat -O on the distributing stack to confirm that there are target stacks available with server applications ready. This display will only show target stacks that are currently up and have joined the sysplex. The READY field indicates how many, if any, applications the target TCP/IP, identified by its DestXCF Addr, has bound to DPort. If none, then this target TCP/IP will not receive any connection workload. The TotalConn field indicates how many connections this distributing TCP/IP has forwarded to the target TCP/IP.

Note: TotalConn is a historical count and will wrap.

The following netstat display command on the distributing stack, TCPCS, shows which target stacks are available with the server applications ready. The target stack is identified by its Dynamic XCF address (DESTXCF ADDR). The READY filed indicates how many application on that target stack have bound to the DPORT. TOTALCONN is the number of all connections the distributing stack, TCPCS, has routed to the target stack. WLM is the Workload Manager weight value for the target TCP/IP stack. WQ is the Workload Manager weight value for the target stack after modification using QoS information provided by the Policy Agent.

```
d tcpip,tcpcs,net,vdpt

EZZ2500I NETSTAT CS V2R10 TCPCS 439
DYNAMIC VIPA DISTRIBUTION PORT TABLE:
DEST IPADDR      DPORT  DESTXCF ADDR     READY       TOTALCONN
201.2.10.11      00020  193.9.200.1      0000000000  0000003561
201.2.10.11      00020  193.9.200.2      0000000000  0000003500
201.2.10.11      00020  193.9.200.3      0000000000  0000003700
201.2.10.11      00021  193.9.200.1      0000000000  0000000499
201.2.10.11      00021  193.9.200.2      0000000000  0000000450
201.2.10.11      00021  193.9.200.3      0000000000  0000000415
201.2.10.12      00020  193.9.200.2      0000000000  0000000239
201.2.10.12      00021  193.9.200.2      0000000000  0000000059
```

The following netstat display command on the distributing stack displays all current connections being distributed by TCPCS.

```
d tcpip,tcpcs,net,vcrt

EZZ2500I NETSTAT CS V2R10 TCPCS 363
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST IPADDR      DPORT  SRC IPADDR     SPORT       DESTXCF ADDR
-----------      -----  ----------     -----       ------------
201.2.10.11      00021  193.9.200.5    01029       193.9.200.1
```

```
201.2.10.11      00021  193.9.200.8    01050       193.9.200.2
201.2.10.11      00021  193.9.200.11   01079       193.9.200.3
201.2.10.12      00021  193.9.200.9    01030       193.9.200.2
```

Dynamic VIPAs and Routing Protocols

OROUTED

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

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

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

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

OMPROUTE

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

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

Ranges of Dynamic VIPA interfaces can be defined using the subnet mask parameter on the OSPF_Interface or Interface statement. The range defined will be all the IP addresses that fall within the subnet defined by the mask and the IP address. The following example defines a range of Dynamic VIPA addresses from 10.138.165.80 to 10.138.165.95:

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

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

For consistency with the VIPARANGE statement in the TCP/IP profile, any value that may fall within the range can be used with the mask to define a range of

Dynamic VIPAs. The interface statement in the following example has the same meaning as the one in the example above:

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

Notes:

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

Chapter 4. Routing

The objective of this chapter is to guide you through the steps required to configure static or dynamic routing and explain how to verify the configuration. The contents of this chapter are based on the assumption that you understand your entire network configuration. It also assumes that you have read and completed all of the verification tasks outlined in previous chapters in this book.

After reading this chapter, you should be able to do the following:

- Configure static or dynamic routing
- PING a remote host by IP address
- Use TRACERTE to determine the path that will be taken to reach a particular destination
- Use NETSTAT to display your routing table
- Use DISPLAY commands to display dynamic routing information.

Note: The definition or modification of an installation's routing configuration should not be performed without a complete understanding of the entire network design.

Routing Terminology

The following list describes some of the more common IP routing-related terms and concepts. If you need more detailed information, refer to *Routing in the Internet* by Christian Huitema.

General Terms

Routing
The process used in an IP network to deliver a datagram to the correct destination.

Static Routing
Routing that is manually configured and does not change automatically in response to network topology changes.

Dynamic Routing
Routing that is dynamically managed by a routing daemon and automatically changes in response to network topology changes.

Routing Daemon
A server process that manages the IP route table.

Autonomous System (AS)
A group of routers exchanging routing information through a common routing protocol. A single AS can represent a large number of IP networks.

Router
A device or host that interprets protocols at the IP layer and forwards datagrams on a path towards their correct destination.

Exterior Gateway Protocol (EGP)
A routing protocol spoken by routers belonging to different Autonomous Systems when those routers are configured to share routing information between Autonomous Systems. This chapter does not discuss exterior gateway routing.

Interior Gateway Protocol (IGP)
A routing protocol spoken by routers belonging to the same Autonomous System. Each AS has a single IGP. A separate AS within a network can be running a different IGP.

Interior Gateway Protocols (IGP)

An interior gateway protocol is a dynamic route update protocol used between dynamic routers that run on TCP/IP hosts within a single Autonomous System. The routers use this protocol to exchange information about which IP routes the IP hosts have knowledge.

Some of the more common interior gateway protocols are:

Routing Information Protocol (RIP)
RIP uses a distance vector algorithm to calculate the best path to a destination based on the number of hops in the path. RIP has several limitations. Some of the limitations which exist in RIP Version 1 are resolved by RIP Version 2.

RIP Version 2
RIP Version 2 extends RIP Version 1. Among the improvements are support for multicasting and variable subnetting. Variable subnetting allows the division of networks into variable size subnets. For example, one route can represent addresses from 9.1.1.0 through 9.1.1.255 (255.255.255.0 subnet) while another can represent addresses from 9.2.0.0 through 9.2.255.255 (255.255.0.0 subnet).

Open Shortest Path First (OSPF)
OSPF uses a link state or shortest path first algorithm. OSPF's most significant advantage compared to RIP is the reduced time needed to converge after a network change. In general, OSPF is more complicated to configure than RIP and might not be suitable for small networks.

Table 11. Interior Gateway Protocol Characteristics

Feature	RIP-1	RIP-2	OSPF
Algorithm	Distance Vector	Distance Vector	Shortest Path First
Network Load (1)	High	High	Low
CPU Processing Requirement (1)	Low	Low	High
IP Network Design Restrictions	Many	Some	Virtually none
Convergence Time	Up to 180 seconds	Up to 180 seconds	Low
Multicast supported (2)	No	Yes	Yes
Multiple equal-cost routes	No (3)	No(3)	Yes

Table 11. Interior Gateway Protocol Characteristics (continued)

Feature	RIP-1	RIP-2	OSPF

Notes:

1. Depends on network size and stability.
2. Multicast saves CPU cycles on hosts that are not interested in certain periodic updates, such as OSPF link state advertisements or RIP-2 routing table updates. Multicast frames are filtered out either in the device driver or directly on the interface card if this host has not joined the specific multicast group.
3. RIP in OMPROUTE allows multiple equal-cost routes only for directly-connected destinations over redundant interfaces. See "Using Static Routing with OMPROUTE" on page 147.

Static vs Dynamic Routing

Whether static or dynamic routing is used, the IP layer routing mechanism is the same. The IP layer routes a packet by searching its routing table for the most specific route known. Route selection occurs in the following order:

1. If a route exists to the destination address (a host route), it is chosen first.
2. If subnet, network, or supernetwork routes exist to the destination, the route with the most specific network mask (the mask with the most bits on) is chosen second.
3. If multicast default routes exist, the one with the most specific multicast address is chosen third.

Default routes are chosen when no other route exists to a destination. Multiple equal-cost routes are allowed for both static and dynamic routing, as depicted in Table 11 on page 144.

The difference between static and dynamic routing is the use of a routing daemon. Dynamic routing allows a TCP/IP stack to respond to network topology changes. Most installations that require robust and reliable networks will use dynamic routing.

The Sample Network

Figure 18 on page 146 shows a network diagram that depicts a sample network. This sample will be used in the following sections as the configuration of static and dynamic routing is described. See "Static Routing" on page 146 and "Dynamic Routing Using OMPROUTE" on page 151 for more information.

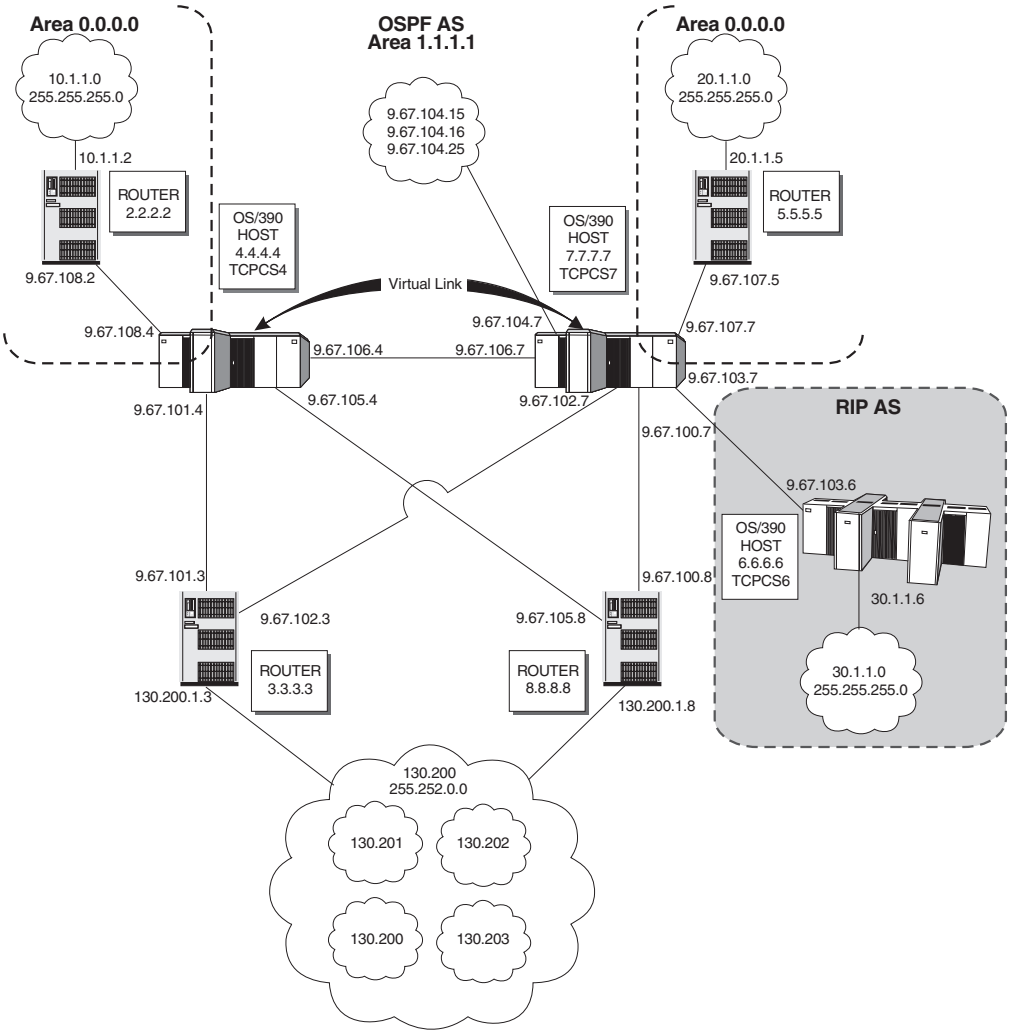


Figure 18. Sample Network

Note: In this sample network, TCPCS4 and TCPCS7 are both performing as OSPF Area Border Routers between OSPF Areas 0.0.0.0 and 1.1.1.1. TCPCS7 is also performing as an AS Boundary Router between the OSPF AS and the RIP AS.

Static Routing

Static routing requires that routes are configured manually for each router; this is a significant reason system administrators avoid this technique (if given a choice). Static routing has the disadvantage that network reachability is not dependent on the state of the network itself. If a destination is down or unreachable via that statically configured route, the static routes remain in the routing table, and traffic continues to be sent toward that destination without success.

To minimize network administrator's tasks, configuration of static routes is avoided, especially in a large network. However, certain circumstances make static routing more appropriate. For example, static routes can be used:

- To define a default route or a route that is not being advertised within a network
- To replace exterior gateway protocols when:
  - Trying to avoid the cost of routing protocol traffic between ASs

– Trying to avoid complex routing policies

If static routing is used, only the PROFILE.TCPIP data set has to be updated with either the BEGINROUTES or GATEWAY statements. Although both statements are functionally equivalent, the BEGINROUTES statement is recommended to define static routes due to its ease of use.

The only ways to modify static routes are:

- Replace the routing table using the VARY OBEY command
- Use incoming ICMP Redirect packets
- Use ICMP Must Fragment packets

See *z/OS Communications Server: IP Configuration Reference* for more information about the VARY OBEY command, IPCONFIG IGNOREREDIRECTS, and IPCONFIG PATHMTUDISC.

Note that the first BEGINROUTES or GATEWAY statement in PROFILE.TCPIP or VARY OBEY replaces all static routes in the TCP/IP stack routing table (including those destination addresses specified in the BSDROUTINGPARMS section of the PROFILE.TCPIP). Subsequent statements within the same data set append to the routing table. Also, both BEGINROUTES and GATEWAY statements cannot be used within the same data set.

Every interface must have an IP address to transmit or receive packets. Along with the IP address, each interface must have a subnet mask associated with it for routing purposes. The combination of the address and mask will yield the subnet that the interface belongs to and also determines the broadcast address for the interfaces. There are two ways to specify the subnet mask:

- Specify the netmask on the BSDROUTINGPARMS statement in PROFILE.TCPIP
- Allow z/OS CS to default the interface netmask

The BSDROUTINGPARMS statement is highly recommended to set the netmask value for *each* physical interface. If either OROUTED or NCPROUTE is used, then BSDROUTINGPARMS are required.

Using Static Routing with OMPROUTE

It is recommended that static routes not be used with OMPROUTE because this will prevent those routes from being dynamically updated in response to network topology changes. An exception is when routes need to be defined to destinations which, for some reason, will not be learned dynamically via the routing protocol. If static routes are required, use the BEGINROUTES or GATEWAY statement in PROFILE.TCPIP to define them.

Another situation where static routes might be required is when multiple, equal-cost routes to a destination are needed and the RIP routing protocol is being used. This is due to the fact that, with the exception of directly attached resources, the RIP protocol will not create multiple, equal-cost routes to a destination. In other words, if multiple adjacent routers are advertising via RIP that they can reach the same destination, RIP will add a route to the TCP/IP route table via only one of those adjacent routers. If it is required that more than one of these routes exist, they would need to be statically configured using the BEGINROUTES or GATEWAY statement in PROFILE.TCPIP. If OROUTED is used instead of OMPROUTED, external entries in the gateways file will be needed. An example of this would be if in Figure 18 on page 146, TCPCS4 used the RIP protocol to Router 3.3.3.3 and Router 8.8.8.8 and if multiple routes were desired to 130.200.0.0 network.

If an installation has multiple interfaces to a directly attached network and it wants to use one interface for input packets and one for output packets (traffic splitting), the installation must use static routes. To do this, a static route could be defined for one and only one interface, forcing all output packets to use that interface. The other routers on the directly attached network would have to be defined with a similar static route, but for the other interface. Although this is the easiest way to implement traffic splitting, if one of the interfaces fails, a host might become unreachable even though the other physical connection may still exist.

Note: A more robust way of accomplishing traffic splitting is to use dynamic routes and make one route preferred over the other via the configured interface costs. See "Step 5: Defining Interface Costs (OSPF and RIP)" on page 170 for more information.

The BSDROUTINGPARMS statement in PROFILE.TCPIP is not used when the OMPROUTE routing daemon is used. Instead, the interface characteristics, including subnet mask, are defined in the OMPROUTE configuration file.

Note: If you are using NCPROUTE with OMPROUTE, the BSDROUTINGPARMS statement is required to route Transport PDUs prior to OMPROUTE activation. Because the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in the BSDROUTINGPARMS statement and the OMPROUTE configuration file.

Static Routing Configuration Examples

The following sections illustrate static routing configuration examples.

OS/390 TCPCS4

BEGINROUTES statements for OS/390 TCPCS4

```
BEGINRoutes      ; first BEGINRoutes in the profile
;       Network/mask    FirstHop    LinkName PacketSize
  Route 9.67.106.0/24   =           CTC4TO7  MTU 1500 ; route1
  Route 9.67.105.0/24   =           CTC4TO8  MTU 1500 ; route2
  Route 9.67.101.0/24   =           CTC4TO3  MTU 1500 ; route3
  Route 9.67.108.0/24   =           CTC4TO2  MTU 1500 ; route4
  Route 9.67.107.0/24   9.67.106.7  CTC4TO7  MTU 1500 ; route5
  Route 7.7.7.7/32      9.67.106.7  CTC4TO7  MTU 1500 ; route6
  Route 9.67.103.0/24   9.67.101.3  CTC4TO3  MTU 1500 ; route7
  Route 9.67.103.0/24   9.67.106.7  CTC4TO7  MTU 1500 ; route8
  Route 30.1.1.0/24     9.67.106.7  CTC4TO7  MTU 1500 ; route9
  Route 10.1.1.0/24     9.67.108.2  CTC4TO2  MTU 1500 ; route10
  Route 130.200.0.0/14  9.67.101.3  CTC4TO3  MTU 1500 ; route11
  Route 130.200.0.0/14  9.67.105.8  CTC4TO8  MTU 1500 ; route12
  Route 130.203.0.0/16  9.67.105.8  CTC4TO8  MTU 1500 ; route13
  Route DEFAULT         9.67.106.7  CTC4TO7  MTU 1500 ; route14
  EndRoutes
;
```

Notes:

1. In the BEGINROUTES block, the netmask can be specified by a '/xx'. This number, denoted by xx, represents the number of significant bits in the netmask. For example:

   ```
   /16 = 16 significant bits = 11111111 11111111 00000000 00000000 = 255.255.0.0
   ```

2. For direct routes, use an equals symbol (=) for the first hop.

GATEWAY statements for OS/390 TCPCS4

```
   ; TCPCS4 Gateway
IPCONFIG VARSUBNET ; Needed for the supernetwork route11 & 12
         DATAGRAMFWD ; Enables data transfer between networks
Gateway          ; first GateWay in the profile
; Network  FirstHop   LnkName   MTU Subnet Mask Subnet Value
  9        =          CTC4TO7  1500 0.255.255.0 0.67.106.0 ; route1
  9        =          CTC4TO8  1500 0.255.255.0 0.67.105.0 ; route2
  9        =          CTC4TO3  1500 0.255.255.0 0.67.101.0 ; route3
  9        =          CTC4TO2  1500 0.255.255.0 0.67.108.0 ; route4
  9        9.67.106.7 CTC4TO7  1500 0.255.255.0 0.67.107.0 ; route5
  7.7.7.7  9.67.106.7 CTC4TO7  1500 HOST                    ; route6
  9        9.67.101.3 CTC4TO3  1500 0.255.255.0 0.67.103.0 ; route7
  9        9.67.106.7 CTC4TO7  1500 0.255.255.0 0.67.103.0 ; route8
  30       9.67.106.7 CTC4TO7  1500 0.255.255.0 0.1.1.0    ; route9
  10       9.67.108.2 CTC4TO2  1500 0.255.255.0 0.1.1.0    ; route10
  130.200  9.67.101.3 CTC4TO3  1500 0.3.0.0     0          ; route11
  130.200  9.67.105.8 CTC4TO8  1500 0.3.0.0     0          ; route12
  130.203  9.67.105.8 CTC4TO8  1500 0                      ; route13
  DEFAULT  9.67.106.7 CTC4TO7  1500 0                      ; route14
;
```

Note: When defining a SUPERNET route in the GATEWAY block, you need to determine the subnet mask. To do this, subtract the actual desired subnet mask (in this case is 255.252.0.0) from the class subnet mask (in this case Class C which is 255.255.0.0) The result (0.3.0.0) is specified as the subnet mask for this supernet route. Because of this confusing notation, the BEGINROUTES statement is recommended over the GATEWAY statement.

BSDROUTINGPARMS statements for OS/390 TCPCS4

```
BSDRoutingParms TRUE ; Shown only for completeness
; Linkname    MTU    Metric    Subnet Mask     Dest Address
  CTC4TO8     1500        0    255.255.255.0   0
  CTC4TO7     1500        0    255.255.255.0   0
  CTC4TO3     1500        0    255.255.255.0   0
  CTC4TO2     1500        0    255.255.255.0   0
  VIPA1A      1500        0    255.255.255.254 0
  EndBSDRoutingParms
;
```

OS/390 TCPCS7

BEGINROUTES statements for OS/390 TCPCS7

```
BEGINRoutes
;       Network/mask    FirstHop    LinkName PacketSize
  Route 9.67.106.0/24   =           CTC7TO4  MTU 1500 ; route1
  Route 9.67.100.0/24   =           CTC7TO8  MTU 1500 ; route2
  Route 9.67.102.0/24   =           CTC7TO3  MTU 1500 ; route3
  Route 9.67.103.0/24   =           CTC7TO6  MTU 1500 ; route4
  Route 9.67.107.0/24   =           CTC7TO5  MTU 1500 ; route5
  Route 4.4.4.4/32      9.67.106.4  CTC7TO4  MTU 1500 ; route6
  Route 10.1.1.0/24     9.67.106.4  CTC7TO4  MTU 1500 ; route7
  Route 20.1.1.0/24     9.67.107.5  CTC7TO5  MTU 1500 ; route8
  Route 30.1.1.0/24     9.67.103.6  CTC7TO6  MTU 1500 ; route9
  Route 130.200.0.0/14  9.67.100.8  CTC7TO8  MTU 1500 ; route10
  Route 130.200.0.0/14  9.67.102.8  CTC7TO3  MTU 1500 ; route11
  Route 130.203.0.0/16  9.67.102.3  CTC7TO3  MTU 1500 ; route12
  Route DEFAULT         9.67.107.5  CTC7TO5  MTU 1500 ; route13
  EndRoutes
```

GATEWAY statements for OS/390 TCPCS7

```
; TCPCS7 GATEWAY
IPCONFIG VARSUBNET ; Needed for the supernetwork route10 & 11
Gateway          ; first GateWay in the profile
```

```
; Network  FirstHop   LnkName   MTU Subnet Mask Subnet Value
  9        =          CTC7TO4  1500 0.255.255.0 0.67.106.0 ; route1
  9        =          CTC7TO8  1500 0.255.255.0 0.67.100.0 ; route2
  9        =          CTC7TO3  1500 0.255.255.0 0.67.102.0 ; route3
  9        =          CTC7TO6  1500 0.255.255.0 0.67.103.0 ; route4
  9        =          CTC7TO5  1500 0.255.255.0 0.67.107.0 ; route5
  4.4.4.4  9.67.106.4 CTC7TO4  1500 HOST                   ; route6
  10       9.67.106.4 CTC7TO4  1500 0.255.255.0 0.1.1.0    ; route7
  20       9.67.107.5 CTC7TO5  1500 0.255.255.0 0.1.1.0    ; route8
  30       9.67.103.6 CTC7TO6  1500 0.255.255.0 0.1.1.0    ; route9
  130.200  9.67.100.8 CTC7TO8  1500 0.3.0.0  0             ; route10
  130.200  9.67.102.8 CTC7TO3  1500 0.3.0.0  0             ; route11
  130.203  9.67.102.3 CTC7TO3  1500 0                      ; route12
  DEFAULT  9.67.107.5 CTC7TO5  1500 0                      ; route13
;
```

BSDROUTINGPARMS statements for OS/390 TCPCS7

```
BSDRoutingParms TRUE
; Linkname    MTU    Metric    Subnet Mask     Dest Address
  CTC7TO8     1500        0    255.255.255.0   0
  CTC7TO3     1500        0    255.255.255.0   0
  CTC7TO6     1500        0    255.255.255.0   0
  CTC7TO4     1500        0    255.255.255.0   0
  CTC7TO5     1500        0    255.255.255.0   0
  VIPA1A      1500        0    255.255.255.254 0
  EndBSDRoutingParms
;
```

Because the sample configuration has a supernet route for 130.200.0.0, IPCONFIG VARSUBNETTING is required in PROFILE.TCPIP. A supernet route means that the netmask for the route is smaller than the class netmask. In this case, 130.200.0.0 is a class B address. The default netmask for class B is 255.255.0.0. The netmask used for this sample is 255.252.0, which is less than 255.255.0.0, hence making this a supernet route. In routing, the stack determines a route that has the most bits in common. Therefore, the stack chooses a route in the following order:

1. HOST
2. SUBNETWORK
3. NETWORK
4. SUPERNETWORK
5. DEFAULT

For example, for TCPCS4 (and when trying to reach 130.200.0.0), route12 in the list is used, which is the supernet route 130.200.0.0 with mask 255.252.0.0. If applying the mask of that route, 255.252.0.0, to the destination IP address, 130.200.0.0, the result is 130.200.0.0 which is the IP address of this route. Now, when trying to reach destination 130.203.5.2, the stack would use route13 in the list which is a network route for 130.203.00 with mask 255.255.0.0. If applying the mask of that route, 255.255.0.0, to the destination IP address, 130.203.5.2, the result is 130.203.00 which is the IP address of this route.

For TCPCS4, route8 and route7 are examples of equal cost multipath routes to get to 9.67.103.0 subnet. This means that TCPCS4 has two different routes to get to this destination. If IPCONFIG MULTIPATH is not enabled, then only route8 will be used as long as it is active. This is because the stack chooses the last route and ignores route7. If route8 becomes inactive, then the stack will switch and use route7. If MULTIPATH is enabled, then the stack will use both routes according to the MULTIPATH specification.

In the example above, all of the links have a subnet mask of 255.255.255.0 because this is what is specified for the links in the BSDROUTINGPARMS. Therefore, to determine the broadcast addresses for link CTC4TO3, AND the IP Address, 9.67.101.4, and the subnet mask, 255.255.255.0, to yield the subnet for this link, 9.67.101.0. Then, OR the subnet, 9.67.101.0, with the complement of the subnet mask, 0.0.0.255. This determines that the broadcast address for this link is 9.67.101.255.

Notes:

1. All IP addresses must follow Classless Inter-Domain Routing (CIDR) convention that requires the actual mask to be one or more on-bits followed by zero or more off-bits. On-bits cannot be followed by off-bits followed by on-bits. Therefore, a class A mask of 255.255.254.0 is valid (an actual mask of FFFFE00), but a class A mask of 255.255.253.0 is not valid (an actual mask of FFFFD00) because 253 is 11111101.
2. VIPA links are not allowed on the GATEWAY or BEGINROUTES statements.
3. You must have a Direct route to a specific IP Address before using that IP Address as the first-hop for indirect routes. A direct route is a route to a destination that is directly connected to the stack by an interface. An indirect route is a route to a destination that is not directly connected, and therefore a router is used to reach that destination. In the example above for TCPCS4, the subnet route for 9.67.101.0 is directly connected to TCPCS4 via link CTC4T03. However, the subnet route for 9.67.103.0 is indirectly connected and the router used to reach that destination is 9.67.106.7 and/or 9.67.101.3, depending on the MULTIPATH definition.

Routing Daemons

Daemon is a UNIX term for a background server process. Daemons are used for dynamic routing. For z/OS CS IP, there are two routing daemons:

OROUTED

OROUTED is an IP routing daemon that implements RIP Version 1 and RIP Version 2. It creates and maintains network routing tables. OROUTED determines if a new route has been established or whether a route is temporarily unavailable. For more information, see “Appendix C. Configuring the OROUTED Server” on page 557

OMPROUTE

OMPROUTE is an IP routing daemon that supports RIP Version 1, RIP Version 2, and OSPF protocols. OMPROUTE is the recommended routing daemon application for z/OS CS IP.

Note: OROUTED and OMPROUTE will not run concurrently on the same TCP/IP stack.

Dynamic Routing Using OMPROUTE

OMPROUTE implements the OSPF protocol described in RFC 1583 (OSPF Version 2), the OSPF subagent protocol described in RFC 1850, 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. The MVS host running with OMPROUTE becomes an active OSPF, RIP router, or in a TCP/IP network. Either or both of these routing protocols can be used to dynamically maintain the host routing table. For example, OMPROUTE can detect when a route

is created, is temporarily unavailable, or if a more efficient route exists. If both OSPF and RIP protocols are used simultaneously, OSPF routes will be preferred over RIP routes to the same destination.

Supported Protocols

Open Shorted Path First (OSPF)

OSPF is classified as an Interior Gateway Protocol (IGP). This means that it distributes routing information between routers belonging to a single Autonomous System (AS), a group of routers all using a common routing protocol. The OSPF protocol is based on link-state or shortest path first (SPF) technology. It has been designed expressly for the TCP/IP internet environment, including explicit support for IP subnetting and the tagging of externally-derived routing information.

OSPF performs the following tasks:

Multiple Routes
Provides support for multiple equal-cost routes

Authentication
Provides for the authentication of routing updates

IP Multicast
Uses IP multicast when sending or receiving the updates

Area Routing Capability
Area routing capability enables an additional level of routing protection and a reduction in routing protocol traffic.

Allows Network Grouping
Allows sets of networks to be grouped together. Such a grouping is called an area. The topology of an area is hidden from the rest of the Autonomous System. This method of hiding information enables a significant reduction in routing traffic. Also, routing within the area is determined only by the area's own topology, lending the area protection from bad routing data. An area is a generalization of an IP subnetted network.

IP Subnet Configuration
Enables the flexible configuration of IP subnets. Each route distributed by OSPF has a destination and mask. Two different subnets of the same IP network number may have different sizes (that is, different masks). This is commonly referred to as variable length subnetting. A packet is routed to the best (longest or most specific) match. Host routes are considered to be subnets whose masks are ″all ones″ (0xFFFFFFFF).

Authenticate OSPF Protocol Exchanges
Can be configured such that all OSPF protocol exchanges are authenticated. This means that only trusted routers can participate in the Autonomous System's routing. A single authentication scheme is configured for each area. This enables some areas to use authentication while others do not.

OSPF is a dynamic routing protocol. It quickly detects topological changes in the AS (such as router interface failures) and calculates new loop-free routes after a period of convergence. This period of convergence is short and involves a minimum of routing traffic as compared to RIP protocol.

In a link-state routing protocol, each router maintains a database describing the Autonomous System's topology. Each participating router has an identical database.

Each individual piece of this database is a particular router's local state (for example, the router's usable interfaces and reachable neighbors). The router distributes its local state throughout the Autonomous System by flooding.

All routers run the exact same algorithm, in parallel. From the topological database, each router constructs a tree of shortest paths with itself as root. This shortest-path tree gives the route to each destination in the Autonomous System. Externally derived routing information appears on the tree as leaves. When several equal-cost routes to a destination exist, the routes (up to four) are added to the TCP/IP stack's route table. The TCP/IP stack uses these equal-cost routes according to the IPCONFIG MULTIPATH statement.

Externally derived routing data (for example, routes learned from the RIP protocol) is passed transparently throughout the Autonomous System. This externally derived data is kept separate from the OSPF protocol's link state data. Each external route can also be tagged by the advertising router, enabling the passing of additional information between routers on the boundaries of the Autonomous System. For information on configuring OSPF, see "Configuring OSPF and RIP" on page 163.

RIP Protocol

RIP is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. RIP is based on the Bellman-Ford or the distance-vector algorithm. RIP has many limitations and is not suited for every TCP/IP environment. Before using the RIP function in OMPROUTE, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about RFCs 1058 and 1723.

RIP uses the number of hops, or hop count, to determine the best possible route to a host or network. The term hop count is also referred to as the metric. In RIP, a hop count of 16 means infinity, or that the destination cannot be reached. This limits the longest path in the network that can be managed by RIP to 15 gateways.

A RIP router broadcasts routing information to its directly connected networks every 30 seconds. It receives updates from neighboring RIP routers every 30 seconds and uses the information contained in these updates to maintain the routing table. If an update has not been received from a neighboring RIP router in 180 seconds, a RIP router assumes that the neighboring RIP router is down and sets all routes through that router to a metric of 16 (infinity). If an update has still not been received from the neighboring RIP router after another 120 seconds, the RIP router deletes from the routing table all of the routes through that neighboring RIP router.

RIP Version 2 is an extension of RIP Version 1 and provides the following features:

Route Tags to provide EGP-RIP and BGP-RIP interactions
: The route tags are used to separate *internal* RIP routes (routes for networks within the RIP routing domain) from *external* RIP routes, which may have been imported from an EGP (external gateway protocol) or another IGP. OMPROUTE does not generate route tags, but preserves them in received routes and readvertises them when necessary.

Variable subnetting support
: Variable length subnet masks are included in routing information so that dynamically added routes to destinations outside subnetworks or networks can be reached.

Immediate Next Hop for shorter paths
: Next hop IP addresses, whenever applicable, are included in the routing

information to eliminate packets being routed through extra hops in the network. OMPROUTE will not generate immediate next hops, but will preserve them if they are included in the RIP packets.

Multicasting to reduce load on hosts
IP multicast address 224.0.0.9, reserved for RIP Version 2 packets, is used to reduce unnecessary load on hosts which are not listening for RIP Version 2 messages. This support is dependent on interfaces that are multicast-capable.

Authentication for routing update security
Authentication keys can be configured for inclusion in outgoing RIP Version 2 packets. Incoming RIP Version 2 packets are checked against the configured keys.

Configuration switches for RIP Version 1 and RIP Version 2 packets
Configuration parameters allow for controlling which version of RIP packets are to be sent or received over each interface.

Supernetting support
The supernetting feature is part of Classless InterDomain Routing (CIDR). Supernetting provides a way to combine multiple network routes into fewer supernet routes, thus reducing the number of routes in the routing table and in advertisements.

For configuration information for RIP, see "Configuring OSPF and RIP" on page 163.

OMPROUTE Configuration

Run-time Environment

OMPROUTE is an z/OS UNIX application, and it requires the Hierarchical File System (HFS) to operate. It can be started from an MVS started procedure, from the z/OS shell, or from AUTOLOG (see "Autolog Considerations for OMPROUTE" on page 157 for restrictions on using AUTOLOG to start OMPROUTE). OMPROUTE must be started by an RACF-authorized user ID, and it must reside in an APF authorized library.

OMPROUTE uses the MVS operator's console, SYSLOGD, CTRACE, and STDOUT for its logging and tracing. The MVS operator's console and SYSLOGD are used for major events such as initialization, termination, and error conditions. CTRACE is used for tracing the receipt and transmission of OSPF/RIP packets as well as communications between OMPROUTE and the TCP/IP stack. STDOUT is used for detailed tracing and debugging.

OMPROUTE uses a standard message catalog. The message catalog must be in the HFS. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.

Configuration of OMPROUTE is via an OMPROUTE configuration file. For details on the statements in the OMPROUTE configuration file, refer to *z/OS Communications Server: IP Configuration Reference*.

Display of OMPROUTE information is performed using the DISPLAY command. Modification of OMPROUTE information is performed using the MODIFY command. For details on OMPROUTE's DISPLAY and MODIFY commands, see *z/OS Communications Server: IP Configuration Reference*.

Multiple TCP/IP Stacks

A one-to-one relationship exists between an instance of OMPROUTE and a stack. OSPF/RIP support on multiple stacks requires multiple instances of OMPROUTE. OMPROUTE and OROUTED cannot run on the same stack concurrently.

TCP/IP Stack Routing Table Management

OMPROUTE's job is limited to the management of the TCP/IP stack routing table. OMPROUTE is not involved in the actual routing decisions made by the TCP/IP stack when routing a packet to its destination.

All dynamic routes are deleted from the stack's routing table upon initialization of OMPROUTE. OMPROUTE then repopulates the stack routing table using information learned via the routing protocols.

ICMP Redirects are ignored when OMPROUTE is active.

Unlike OROUTED, OMPROUTE does not make use of the BSDROUTINGPARMS 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 OMPROUTE configuration file.

Using RIP and OSPF with OMPROUTE

When OMPROUTE is initialized, it uses the OMPROUTE configuration file to determine which routing protocols will be enabled. If at least one OSPF interface is configured, the OSPF protocol is enabled. If at least one RIP interface is configured, RIP is enabled. If OMPROUTE is started with no interfaces defined for a particular protocol, that protocol is disabled until one of the following occurs:

- OMPROUTE is stopped and restarted with a configuration file containing at least one interface of the specific type
- OMPROUTE is dynamically reconfigured via the MODIFY command with a configuration file containing at least one interface of the specific type

When OMPROUTE is configured for both the OSPF and RIP protocols, routes that are learned through the OSPF protocol take precedence over routes learned through the RIP protocol.

The OSPF and RIP protocols are communicated over interfaces that are defined with the OSPF_INTERFACE and RIP_INTERFACE configuration statements, respectively. An interface involved in the communication of neither the RIP nor the OSPF protocol should be configured to OMPROUTE via the INTERFACE configuration statement. For non-point-to-point interfaces, an INTERFACE statement is required only to change the default values used by OMPROUTE (for example, to change the default MTU.) Refer to "VIPA Interfaces (Static VIPA and Dynamic VIPA)" on page 168 for special VIPA consideration.

OMPROUTE allows for the generation of multiple, equal-cost routes to a destination. For OSPF and RIP, up to four multiple equal-cost routes are allowed. For RIP, multiple equal-cost routes are supported only to directly connected destinations over redundant interfaces.

Special Considerations

Token-Ring Multicast: If OMPROUTE will be communicating through the OSPF or RIP Version 2 protocol over a token ring media, and there will be routers

attached to that token ring that are not listening (at the DLC layer) for the token ring multicast MAC address 0xC000.0004.0000, the following TRANSLATE statement is required in the PROFILE.TCPIP:

```
TRANSLATE 224.0.0.0 IBMTR FFFFFFFFFFFF linkname
```

Without this statement, OSPF and RIP Version 2 multicast packets are discarded at the DLC layer by those routers that are not listening for the token ring multicast MAC address.

Virtual IP Addresses (VIPA): OMPROUTE is enhanced with Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. The VIPA routes are included in the OSPF and RIP advertisements to adjacent routers. Adjacent routers learn about VIPA routes from the advertisements and can use them to reach the destinations at the MVS host.

Service Policy: If service policy is going to be used to restrict access to neighbors on point-to-multipoint interfaces (for example MPCPTP interfaces including XCF and IUTSAMEH connections) for temporary intervals, those neighbors must be explicitly defined on the OSPF_INTERFACE or RIP_INTERFACE statement. Otherwise, OMPROUTE might not be able to communicate with those neighbors when the access restriction expires.

Multiple Equal-Cost Routes: When IPCONFIG MULTIPATH is specified in PROFILE.TCPIP and multiple routes exist in the TCP/IP route table for a destination, outbound traffic for that destination will be spread across all of the routes. This traffic spreading will be done on either a packet-basis or connection-basis depending on the parameter specified on IPCONFIG MULTIPATH. When OMPROUTE is being used to provide dynamic routing for a TCP/IP stack, multiple routes to the same destination can be dynamically added to the TCP/IP stack's route table, based upon the routing information learned from other routers. These multiple routes will be added when the route calculation for each has resulted in the same route cost value. No more than four equal-cost routes will be added for each destination. For RIP, multiple equal-cost routes will be added only to directly-connected destinations over redundant interfaces. The RIP protocol will generate no more than one indirect route to a destination.

Configuring OMPROUTE

The steps to configure OMPROUTE are:

1. Create the OMPROUTE configuration file.
2. Reserve the RIP UDP port (if using the RIP protocol).
3. Update the resolver configuration file.
4. Update the OMPROUTE cataloged procedure.
5. Specify the RIP UDP port number in the SERVICES file or data set (if using the RIP protocol).
6. RACF authorize user IDs for starting OMPROUTE.
7. Start syslogd.
8. Update the OMPROUTE environment variables (optional).
9. Create static routes (optional).

These steps are described in the following sections.

Step 1: Create the OMPROUTE Configuration File

The OMPROUTE configuration file provides information about the host's routing capabilities and TCP/IP interfaces. See "Configuring OSPF and RIP" on page 163

for more detail about the contents of this file. The following is the search order used by OMPROUTE to locate the configuration data set or file:

1. If the environment variable, OMPROUTE_FILE, has been defined, OMPROUTE uses the value as the name of an MVS data set or HFS file to access the configuration data. The syntax for an MVS data set name is `//mvs.dataset.name`. The syntax for an HFS file name is `/dir/subdir/file.name`.
2. `/etc/omproute.conf`
3. hlq.ETC.OMPROUTE.CONF

A sample configuration file is provided in SEZAINST(EZAORCFG). The configuration file for TCPCS4, TCPCS6, and TCPCS7 in the sample network are shown in Figure 18 on page 146. For a description of the syntax rules for the OMPROUTE configuration file, as well as details on each of the configuration statements, refer to the *z/OS Communications Server: IP Configuration Reference*.

Step 2: Reserve the RIP UDP Port (If Using the RIP Protocol)

If the RIP protocol of OMPROUTE is going to be used, UDP port 520 should be reserved for OMPROUTE. This is done by adding the name of the member containing the OMPROUTE cataloged procedure to the PORT statement in PROFILE.TCPIP:

```
PORT
    520 UDP OMPROUTE
```

If you want to be able to start OMPROUTE from the z/OS shell, use the special name OMVS as follows:

```
 PORT
    520 UDP OMVS
```

Autolog Considerations for OMPROUTE: As discussed in *z/OS Communications Server: IP Configuration Reference*, if a procedure in the AUTOLOG list also has a PORT statement reserving a TCP or UDP port but does not have a listening connection on that port, TCP/IP periodically attempts to cancel that procedure and start it again.

Therefore, if OMPROUTE is being started with AUTOLOG and only the OSPF protocol is being used (no RIP protocol and, therefore, no listening connection on the RIP UDP port), it is important to do one of the following:

- Ensure that the RIP UDP port (520) is not reserved by the PORT statement in the PROFILE.TCPIP.
- Add the NOAUTOLOG parameter to the PORT statement in the PROFILE.TCPIP. For example,

  ```
  PORT
   520 UDP OMPROUTE 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, OMPROUTE will be periodically cancelled and restarted by TCP/IP.

Step 3: Update the Resolver Configuration File

The resolver configuration file contains keywords (DATASETPREFIX and TCPIPjobname) used by OMPROUTE. The value assigned to DATASETPREFIX will determine the high-level qualifier (hlq). The hlq is used in the search order for the OMPROUTE configuration file. If no DATASETPREFIX keyword is found, a default of TCPIP is used. The value assigned to TCPIPjobname will be used as the name of the TCP/IP stack with which OMPROUTE establishes a connection.

For a description of the search order used by the resolver to locate the resolver configuration file, see “Resolver Configuration Files” on page 15.

Step 4: Update the OMPROUTE Cataloged Procedure

If OMPROUTE is to be started by a procedure, create the cataloged procedure by copying the sample in SEZAINST(OMPROUTE) to your system or recognized PROCLIB. Specify OMPROUTE parameters and change the data set names to suit your local configuration.

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZBORPRC
//*
//*        5647-A01 (C) Copyright IBM Corp. 1998.
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//OMPROUTE PROC
//OMPROUTE EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/')
//*
//* Example of start parameters to OMPROUTE:
//*
//* PARM=('POSIX(ON)',
//*       'ENVAR("_CEE_ENVFILE=DD:STDENV")/-t1')
//*
//*        Provide environment variables to run with the
//*        desired stack and configuration.  As an example,
//*        the file specified by STDENV could have these
//*        four lines in it:
//*
//*           RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*           OMPROUTE_FILE=/u/usernnn/config.tcpcs2
//*           OMPROUTE_DEBUG_FILE=/tmp/logs/omproute.debug
//*           OMPROUTE_DEBUG_CONTROL=1000,5
//*
//*        For information on the above environment variables,
//*        refer to the IP CONFIGURATION GUIDE.
//*
//STDENV   DD PATH='/u/usernnn/envcs2',
//         PATHOPTS=(ORDONLY)
//*
//*        The stdout stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stdout file
//*        every time OMPROUTE is started.  If you want to retain
//*        previous stdout information, change it to OAPPEND.
//*
//SYSPRINT DD SYSOUT=*
//*SYSPRINT DD PATH='/tmp/omproute.stdout',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
```

```
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*        The stderr stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stderr file
//*        every time OMPROUTE is started.  If you want to retain
//*        previous stderr information, change it to OAPPEND.
//*
//SYSOUT   DD SYSOUT=*
//*SYSOUT   DD PATH='/tmp/omproute.stderr',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Step 5: Specify the RIP UDP Port Number in the SERVICES File (If Using the RIP Protocol)

The services file contains the relationship between services and port numbers as described in *z/OS Communications Server: IP Configuration Reference*. The portion of the services file relevant to OMPROUTE is:

```
route           520/udp         router routed
```

The file must exist for the RIP protocol of OMPROUTE to operate.

For a description of the search order used to locate the services file, see "ETC.SERVICES" on page 21.

Step 6: RACF-Authorize User IDs for Starting OMPROUTE

To reduce risk of an unauthorized user starting OMPROUTE and affecting the contents of the routing table, users who start OMPROUTE must be RACF-authorized to the entity MVS.ROUTEMGR.OMPROUTE. To do this, the following commands must be entered from an RACF user ID, substituting the authorized user ID on the ID (userid) parameter. The commands in the following example are taken from SEZAINST(EZARCF).

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OMPROUTE) UACC(NONE)
PERMIT MVS.ROUTEMGR.OMPROUTE ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Step 7: Start syslogd

To write only the urgent OMPROUTE messages to the z/OS console, syslogd should be running while OMPROUTE is running. Syslogd sends the non-urgent messages to the HFS message log.

Step 8: Update the OMPROUTE Environment Variables (Optional)

The following environment variables are used by OMPROUTE and can be tailored to a particular installation:

RESOLVER_CONFIG

The RESOLVER_CONFIG variable is used by OMPROUTE to locate the resolver configuration file. For more information on OMPROUTE's use of the resolver configuration file, see "Step 3: Update the Resolver Configuration File" on page 158. For more information about the RESOLVER_CONFIG environment variable, refer to *z/OS UNIX System Services Planning*.

OMPROUTE_FILE

The OMPROUTE_FILE variable is used by OMPROUTE in the search order for the OMPROUTE configuration file. For details on the search order

used for locating this configuration file, see “Step 1: Create the OMPROUTE Configuration File” on page 156.

OMPROUTE_DEBUG_FILE
The OMPROUTE_DEBUG_FILE variable is used by OMPROUTE to override the debug output destination. For more information on using this environment variable, see “OMPROUTE Parameters” on page 161.

OMPROUTE_DEBUG_FILE_CONTROL
The OMPROUTE_DEBUG_FILE_CONTROL variable is used by OMPROUTE to control the size and quantity of trace files created when the OMPROUTE_DEBUG_FILE variable is specified. The syntax of this variable is:

```
OMPROUTE_DEBUG_FILE_CONTROL=<size of file>,<num of files>
```

The default values for <size of file> and <num of files> are 200, 5 respectively. In general, these values are sufficient for most installation.

Step 9: Create Static Routes (Optional)

OMPROUTE does not use the environment variable GATEWAYS_FILE to initialize static routes. To create static routes, use the BEGINROUTES or GATEWAY statement in PROFILE.TCPIP. For information on the syntax of these statements, see *z/OS Communications Server: IP Configuration Reference*.

During initialization, OMPROUTE learns of static routes by reading the internal routing table set up by TCP/IP.

Note: The use of static routes with OMPROUTE is not recommended. See “Using Static Routing with OMPROUTE” on page 147 for more information.

Starting and Controlling OMPROUTE

After the necessary RACF authorization has been defined (see “Step 6: RACF-Authorize User IDs for Starting OMPROUTE” on page 159), OMPROUTE can be started from an MVS procedure, from the z/OS shell, or from AUTOLOG.

- You can start OMPROUTE from the MVS operators console by starting the OMPROUTE start procedure. A sample start procedure is provided with the product in *hlq*.SEZAINST(OMPROUTE).
- You can start OMPROUTE from the z/OS shell by starting OMVS and then issuing the OMPROUTE command and, optionally, any parameters. For information on parameters, see “OMPROUTE Parameters” on page 161.
- You can use the AUTOLOG statement to start OMPROUTE automatically during TCP/IP initialization. Insert the name of the OMPROUTE start procedure in the AUTOLOG statement of the PROFILE.TCPIP data set.

```
AUTOLOG
  OMPROUTE
ENDAUTOLOG
```

Note: For special considerations when using AUTOLOG to start OMPROUTE, see “Autolog Considerations for OMPROUTE” on page 157.

In a Common INET environment, OMPROUTE will attempt to connect to a stack whose name is determined by the TCPIPjobname keyword from the resolver configuration data set or file. In configurations with multiple stacks, a copy of OMPROUTE must be started for each stack that requires OMPROUTE services. To

associate OMPROUTE with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique TCPIPjobname.

When running from an MVS procedure, the environment variables can be set by using the STDENV DD statement in the OMPROUTE procedure. For information concerning the environment variables used by OMPROUTE, refer to *z/OS Communications Server: IP Configuration Reference*.

OMPROUTE Parameters

OMPROUTE accepts three command line parameters, which govern tracing and debug information. OMPROUTE's trace and debug information is written to stdout with two exceptions:

- When the routing application was started with no tracing, and then a MODIFY command is issued to enable tracing. In this case, the output destination defaults to the file omproute_debug in the current temporary directory (the default is /tmp).
- When the debug output destination has been overridden via the use of an environment variable (OMPROUTE_DEBUG_FILE).

If OMPROUTE is to be started from an MVS procedure, add your parameters to PARM=() in the OMPROUTE cataloged procedure. For example:

```
//* PARM=('POSIX(ON)',
//*       'ENVAR("_CEE_ENVFILE=DD:STDENV")/-t1')
//*
```

If OMPROUTE is to be started from an z/OS shell command line, enter the parameters on the command line.

For either method of starting OMPROUTE, parameters can be specified in mixed case.

Note: Use of the *-tn*, *-dn*, and *-sn* parameters affects OMPROUTE performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

The -tn Command Line Parameter: The *-tn* option specifies the external tracing level, where *n* is a supported trace level. It is intended for customers, testers, service, or developers, and provides information on the operation of the routing application. This option can be used for many purposes, such as debugging a configuration, education on the operation of the routing application, verification of test cases, and so on. The following levels are supported:

1 Informational messages

2 Formatted packet trace

These option levels are cumulative—level 2 includes level 1. For example, -t2 provides formatted packet trace and informational messages.

The -dn and -sn Command Line Parameters: These options specify the internal debugging levels. They are intended for service and provide internal debugging information needed for debugging problems. Use of these parameters can significantly impact performance and are not recommended unless needed to debug a problem. For more information about the use of these parameters, refer to *z/OS Communications Server: IP Diagnosis*.

Controlling OMPROUTE

You can control OMPROUTE from the operator's console using the MODIFY command. The syntax of the MODIFY command can be found in *z/OS Communications Server: IP Configuration Reference*. MODIFY commands are available to perform the following functions:

- "Stopping OMPROUTE"
- "Rereading the Configuration File"
- "Enabling or Disabling the OMPROUTE Subagent"
- "Changing the Cost of OSPF Links" on page 163
- "Controlling OMPROUTE Tracing and Debugging" on page 163

Stopping OMPROUTE: OMPROUTE can be stopped in several ways:

- From MVS, issue STOP <procname> or MODIFY <procname>,KILL.

  If OMPROUTE was started from a cataloged procedure, procname is the member name of that procedure. If OMPROUTE was started from the z/OS shell, procname is useridX, where X is the sequence number set by the system. To determine the sequence number, from the SDSF LOG window on TSO, issue ″/d omvs,u=userid″. This will show the programs running under this user ID. The procname can also be set using the environment variable _BPX_JOBNAME and then starting OMPROUTE in the shell background.
- From a z/OS shell superuser ID, issue the kill command to the process ID (PID) associated with OMPROUTE. To determine the PID, use one of the following methods:
  - From the MVS console, issue D OMVS,U=userid, or issue /D OMVS,U=userid at the SDSF LOG window on TSO (where userid is the user ID that started omproute from the shell).
  - Issue the ps -ef command from the z/OS shell.
  - Record the PID when you start OMPROUTE.

For information on the environment variable _BPX_JOBNAME, refer to *z/OS UNIX System Services Planning*. For information on the D OMVS,U=userid command, refer to *z/OS MVS System Commands*.

Rereading the Configuration File: The MODIFY <procname>,RECONFIG command is used to reread the OMPROUTE configuration file. This command ignores all statements in the configuration file except new OSPF_INTERFACE, RIP_INTERFACE, and INTERFACE statements. These new configuration statements must be reread from the configuration file through this command prior to the interface being configured to the TCP/IP stack.

Enabling or Disabling the OMPROUTE Subagent: Use the MODIFY <procname>,ROUTESA=ENABLE command or the MODIFY <procname>,ROUTESA=DISABLE command to enable or disable the OMPROUTE subagent.

Note: To change any other value on the ROUTESA_CONFIG statement, the OMPROUTE application must be recycled.

The OMPROUTE subagent implements RFC 1850 for the OSPF Protocol. The ROUTESA_CONFIG statement is used in the OMPROUTE configuration file to configure the OMPROUTE subagent. For details on ROUTESA_CONFIG, refer to *z/OS Communications Server: IP Configuration Reference*.

Changing the Cost of OSPF Links: The cost of an OSPF interface can be dynamically changed using the MODIFY <procname>,OSPF,WEIGHT,NAME=<if_name>,COST=<cost> command. This new cost is flooded quickly throughout the OSPF routing domain, and modifies the routing immediately.

The cost of the interface reverts to its configured value whenever the router is restarted. To make the cost change permanent, you must reconfigure the appropriate OSPF_INTERFACE statement in the configuration file.

Controlling OMPROUTE Tracing and Debugging: The following commands are used to start, stop, or change the level of OMPROUTE tracing and debugging:

- MODIFY <procname>,TRACE=n : for OMPROUTE tracing; n can be 0–2
- MODIFY <procname>,DEBUG=n : for OMPROUTE debugging; n can be 0–4
- MODIFY <procname>,SADEBUG=n : for OMPROUTE subagent debugging; n can be 0 or 1

Note: Use of OMPROUTE tracing and debugging affects OMPROUTE performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

Configuring OSPF and RIP

Step 1: Setting the OSPF Router ID (If OSPF Protocol is Used)

Every router in an OSPF Autonomous System must be assigned a unique router ID. The ROUTERID configuration statement should be coded within the OMPROUTE configuration file to assign the router ID. The value must be one of the interface addresses configured to the stack. If the ROUTERID configuration statement is not coded, OMPROUTE chooses the IP address from one of the OSPF_INTERFACE statements as the router ID. With the advent of Dynamic VIPAs (DVIPAs) that can move between z/OS hosts within a sysplex, it is highly recommended that the ROUTERID be a physical interface or a static VIPA, not a Dynamic VIPA.

In the example network shown in Figure 18 on page 146, the ROUTERID is set to the static VIPA address that represents each OMPROUTE router. TCPCS4 has ROUTERID=4.4.4.4, and TCPCS7 has ROUTERID=7.7.7.7.

Step 2: Defining OSPF Areas (If OSPF Protocol is Used)

The sample network shown in Figure 18 on page 146 depicts a network divided using two different methods. The first division is between IP subnetworks within the OSPF Autonomous System (AS) and IP subnetworks external to the OSPF AS (those within the RIP AS). The subnetworks included within the OSPF AS are further 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 compute routes to destinations in different areas. Overhead is reduced because less information is exchanged and stored by routers and because fewer CPU cycles are required for a less complex route table calculation.

Every OSPF AS 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 areas. See “Step 6: Configuring Virtual Links (If OSPF Protocol is Used)” on page 171 for more information on this subject.

Routers that attach to more than one area function as Area Border Routers. All Area Border Routers are part of the backbone, so they must either attach directly to a backbone IP subnet or be connected to another backbone router over a virtual link.

The information and algorithms used by OSPF to calculate routes vary according to whether the destination is within the same area, in a different area within the OSPF AS, or external to the OSPF AS. Every router maintains a database of all links within its area. A shortest path first algorithm is used to calculate the best routes to destinations within the area from this database. Routes between areas are calculated from summary advertisements originated by Area Border Routers for destinations located in other areas of the OSPF AS. External routes (for example, routes to destinations that lie within a RIP AS) are calculated from AS External advertisements originated by AS Boundary Routers and flooded throughout the OSPF AS.

Use the AREA configuration statement to define the areas to which a router attaches. If you do not use the AREA statement, the default is that all OSPF interfaces attach to the backbone area. In the sample network, TCPCS4 and TCPCS7 are both Area Border Routers belonging to both the backbone area (0.0.0.0) and area 1.1.1.1.

```
AREA
  Area_Number=0.0.0.0;

AREA
  Area_Number=1.1.1.1;
```

Step 3: Limiting Information Exchange between OSPF Areas (If OSPF Protocol is Used)

When Area Border Routers are configured, parameters on the AREA and RANGE configuration statements can be used to control the OSPF route information that crosses the area boundary.

One option is to use the AREA statement to define an area as a stub area. AS External 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. Destinations external to the stub area are still reachable due to the Area Border Routers advertising default routes into stub areas. Traffic within the stub area for unknown destinations is forwarded to the Area Border Router (using the default route). The border router uses its more complete routing information to forward the traffic on an appropriate path toward its destination.

The following requirements must be met for an area to be defined as a stub area:

- No virtual links are configured through the area to maintain backbone connectivity.
- It is acceptable for routers within the area to use a default route for traffic destined outside the AS.
- No routers within the area are AS boundary routers (OSPF routers that advertise routes from external sources as AS External advertisements).

The following AREA statement example meets these requirements:

```
AREA
   Area_Number=2.2.2.2
   Stub_area=Yes
   Import_Summaries=No;
```

Another option is to use IP subnet address ranges to limit the number of summary advertisements originated into an area. A range is defined by an IP address and an address mask. Destinations are considered to fall within the range if the destination address and the range IP address match after the range mask has been applied to both addresses.

When a range is configured for an area at an Area Border Router, the border router suppresses summary advertisements for destinations within that area that fall within the range. The suppressed advertisements would have been originated into the other areas 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.

Notes:

1. If the range is not advertised, there will be no inter-area routes for any destination that falls within the range.
2. Ranges cannot be used for areas through which virtual links are configured to maintain backbone connectivity.

In the sample network shown in Figure 18 on page 146, the following RANGE statement could be configured on TCPCS7 to prevent TCPCS7 from advertising destinations in the 9.67.101.0 subnet into the backbone area (Area 0.0.0.0):

```
RANGE
   IP_Address=9.67.101.0
   Subnet_Mask=255.255.255.0
   Area_Number=1.1.1.1
   Advertise=No;
```

Step 4: Defining Interfaces (OSPF and RIP)

Each interface in use by the stack should be defined to OMPROUTE using an OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE statement. This section describes the differences between interface types that you should consider when configuring interfaces to OMPROUTE. In general, use the following guidelines:

- An interface over which the OSPF protocol is communicated with other routers must be configured with the OSPF_INTERFACE statement.
- An interface over which the RIP protocol is communicated with other routers must be configured with the RIP_INTERFACE statement.
- All other interfaces should be configured with the INTERFACE statement.

A VIPA interface is an exception to these guidelines and is discussed in more detail in “VIPA Interfaces (Static VIPA and Dynamic VIPA)” on page 168.

Point-to-Point (For Example CTC and CLAW)

For point-to-point interfaces, the destination IP address must be known to OMPROUTE. Specify the DESTINATION_ADDR parameter to allow for the creation of a host route to the address at the remote end of the interface.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.67.106.7
Name=CTC7TO4
Subnet_mask=255.255.255.0
Attaches_to_Area=1.1.1.1
Destination_Addr=9.67.106.4;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.67.103.7
Name= CTC7TO6
Subnet_mask=255.255.255.0
Destination_Addr=9.67.103.6
RIPV2=Yes;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.67.111.1
Name=CTCX
Subnet_mask=255.255.255.0
Destination_addr=9.67.111.2;
```

Note: If another router is directly attached via a CLAW device, and the OSPF protocol is being communicated with that router, the other router must also be configured to view the CLAW device as a point-to-point interface. Failure to do this results in a failure to add any routes via that router.

Point-to-Multipoint

For Point-to-Multipoint capable interfaces (for example MPCPTP interfaces including XCF and IUTSAMEH connections), OMPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate the OSPF or RIP packets. However, due to underlying signaling that takes place when a host connects to these network types, the stack is able to learn the required addresses. In turn, OMPROUTE learns those IP address from the stack. As a result, it is not necessary to configure the IP addresses of the other routers on the interface statements.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.27.13.81
Name=XCFD00
   Attaches_to_Area=1.1.1.1
Subnet_mask=255.255.255.0;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.27.23.81
Name=MPCA01
Subnet_mask=255.255.255.0
RIPV2=Yes;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.27.33.81
Name=XCFB00
Subnet_mask=255.255.255.0;
```

Non-Broadcast Network Interfaces (For example, Hyperchannel and ATM)

If the OSPF or RIP protocol communicates with one or more routers over a non-broadcast network interface, OMPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate. 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 OMPROUTE with the parameters configured as follows:

- DR_NEIGHBOR and/or the NO_DR_NEIGHBOR parameters on the OSPF_INTERFACE statement
- NEIGHBOR parameter on the RIP_INTERFACE statement
- NON_BROADCAST=YES and ROUTER_PRIORITY parameters on the OSPF_INTERFACE statement

In the OSPF case, DR_NEIGHBOR defines which routers within the non-broadcast network can become the designated router. NO_DR_NEIGHBOR defines which routers cannot become the designated router. ROUTER_PRIORITY defines the priority of this router on the non-broadcast network so that the designated router can be elected for the network. Note that multiple DR_NEIGHBOR and NO_DR_NEIGHBOR parameters can be coded on one statement.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.37.84.49
Name=HCHE00
Subnet_mask=255.255.255.0
   Attaches_to_Area=1.1.1.1
Non_Broadcast=Yes
DR_Neighbor=9.37.84.53
No_DR_Neighbor=9.37.84.63
Cost0=3
Router_Priority=2;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.37.104.79
Name=ATME00
Subnet_mask=255.255.255.0
RIPV2=Yes
Neighbor=9.37.104.85
Neighbor=9.37.104.53;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.77.13.49
Name=ATMB00
Subnet_mask=255.255.255.0;
```

Broadcast Network Interfaces (For Example, Token Ring, Ethernet, and FDDI)

When the OSPF or RIP protocol is communicated over a broadcast medium such as Token Ring, Ethernet, or FDDI, these networks allow for broadcasting and multicasting. Therefore, it is not necessary for OMPROUTE to know the IP addresses of the other routers on the network for OSPF or RIP packets to be communicated with those routers. OMPROUTE sends packets to the other routers on the network by using appropriate broadcast or multicast addresses. The IP addresses of the other routers are learned as OSPF/RIP packets are received from

them. The OSPF_INTERFACE must include the ROUTER_PRIORITY parameter to assist in electing a Designated Router for the network.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.59.101.5
Name=TR1
   Subnet_mask=255.255.255.0
Attaches_to_Area=1.1.1.1
Cost0=2
Router_Priority=1;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.29.107.3
Name=TR2
Subnet_mask=255.255.255.0
RIPV2=Yes;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.77.14.49
Name=ETHB00
Subnet_mask=255.255.255.0;
```

If OMPROUTE will be communicating with the OSPF or RIP Version 2 protocol over a token ring media where an attached router does not listen for multicast MAC address 0xC000.0004.0000, see “Token-Ring Multicast” on page 155.

For interfaces into broadcast media which contain routers that do not support multicast, it is possible to configure the interfaces as Non-Broadcast Network Interfaces. This would cause OMPROUTE to unicast to the neighbor addresses rather than using a multicast address. However, it would also be necessary to configure all the routers on the network to unicast. Otherwise, their multicast packets would never be received.

VIPA Interfaces (Static VIPA and Dynamic VIPA)

If only the RIP protocol is used by OMPROUTE, VIPA interfaces should be defined with the INTERFACE statement. If only OSPF or if both OSPF and RIP are used by OMPROUTE, VIPA interfaces should be defined with the OSPF_INTERFACE statement.

Sample OSPF_INTERFACE

```
OSPF example:
OSPF_INTERFACE
IP_Address=4.4.4.4
Name=VIPA1
   Subnet_mask=255.255.255.254;
```

Sample INTERFACE

```
non-OSPF example:
INTERFACE
IP_Address=6.6.6.6
Name=VIPA1
Subnet_mask=255.255.255.254;
```

Note: The most specific subnet mask you can specify is 255.255.255.254.

For Dynamic VIPA (DVIPA), link names are assigned programmatically by the stack when the DVIPA is created. Therefore, the name field set on the INTERFACE or OSPF_INTERFACE statement is ignored by OMPROUTE for DVIPAs.

Because a stack could have a large number of DVIPAs defined, as well as DVIPA ranges, additional wildcard capabilities exist on the OSPF_INTERFACE and INTERFACE statements for use only with DVIPAs.

Ranges of DVIPA interfaces can be defined using the Subnet_Mask parameter on the OSPF_INTERFACE or INTERFACE statement. The range defined in this way will be all the IP addresses that fall within the subnet defined by the mask and the IP address.

In the example below, DVIPA interfaces in the range of 10.138.65.80 through 10.138.65.95 are defined:

Sample OSPF_INTERFACE

```
OSPF example:
OSPF_INTERFACE
IP_Address=10.138.65.80
Name=DVIPAs
   Subnet_mask=255.255.255.240;
```

Sample INTERFACE

```
non-OSPF example:
INTERFACE
IP_Address=10.138.65.80
Name=DVIPAs
Subnet_mask=255.255.255.240;
```

You must consider an additional issue when VIPAs are being moved between TCP/IP stacks and dynamic routing is provided for those stacks by OMPROUTE. This movement of VIPAs can be done manually or automatically via the use of Dynamic VIPAs. For the VIPAs to be correctly processed and advertised by the routing protocols, they (like all other interfaces) must be configured to OMPROUTE at the time that they become active on the TCP/IP stack. This configuration of VIPAs to OMPROUTE can be accomplished by:

- Explicitly configuring each VIPA with its own OSPF_INTERFACE or INTERFACE statement
- Configuring a range of DVIPAs with a single OSPF_INTERFACE or INTERFACE statement, using the method described above
- Configuring a group of VIPAs with a single OSPF_INTERFACE or INTERFACE statement, using the wildcarding feature available on the interface statements

The recommended approach for configuring OMPROUTE for VIPAs that might move is to preconfigure the OMPROUTE on each TCP/IP stack with all VIPAs that could potentially exist on that stack at some time. Pre-configuring in this way prepares each OMPROUTE for the possible addition of the VIPAs to its stack. During times when the VIPAs do not exist on a particular OMPROUTE's stack, the configuration information will not be used. However, during periods when the VIPAs do exist on that OMPROUTE's stack, the configuration information will be available for use by OMPROUTE. This method is recommended because of its ability to respond to movement of the VIPAs between TCP/IP stacks without modification of the OMPROUTE configuration with each move.

If the pre-configuration of VIPAs described in this section has not been done, it is still possible to define a VIPA to OMPROUTE such that it is properly processed and advertised when it becomes active on the corresponding TCP/IP stack. To do this, add the appropriate OSPF_INTERFACE or INTERFACE statement to the OMPROUTE configuration file and then cause OMPROUTE to reread the configuration file by issuing the MODIFY <procname>,RECONFIG command.

Note: You must modify the OMPROUTE configuration file and issue the RECONFIG command prior to the movement of the VIPA to the corresponding TCP/IP stack.

Step 5: Defining Interface Costs (OSPF and RIP)

Both the OSPF and RIP protocols have a cost value associated with interfaces. With both protocols, the cost of a route to reach a destination is the sum of the costs of each link that will be traversed on the way to the destination. In the sample network shown in Figure 18 on page 146, the cost of a route to get from TCPCS7 to router 3.3.3.3 via TCPCS4 is the cost of the link from TCPCS7 to TCPCS4 plus the cost of the link from TCPCS4 to router 3.3.3.3.

The method for configuring cost values differs between the OSPF and RIP protocols. The cost values of OSPF links, set using the COST0 parameter of the OSPF_INTERFACE statement, should be configured to ensure that preferred routes to destinations will have a lower cost than less preferable routes. The less preferable routes, with the higher cost, will not be used except upon failure of the preferred routes.

For the purpose of the following example, the sample network Figure 18 on page 146 is used and the convention stack (interface) is used to refer to the cost configured for a particular interface on a stack. For instance TCPCS7(9.67.106.7) refers to the cost configured for interface 9.67.106.7 on TCPCS7.

There are three possible routes from TCPCS7 to router 3.3.3.3. They are:

- Direct (TCPCS7 —> 3.3.3.3),
- Via TCPCS4 (TCPCS7 —> TCPCS4 —> 3.3.3.3)
- Via router 8.8.8.8 and TCPCS4 (TCPCS7 —> 8.8.8.8 —> TCPCS4 —> TCPCS3)

If the preferred route from TCPCS7 to router 3.3.3.3 is via TCPCS4, then interface costs must be configured such that the following are true:

```
TCPCS7(9.67.106.7) + TCPCS4(9.67.101.4) < TCPCS7(9.67.102.7)
TCPCS7(9.67.106.7) + TCPCS4(9.67.101.4) < TCPCS7(9.67.100.7) +
8.8.8.8(9.67.105.8) + TCPCS4(9.67.101.4)
```

The reasons for preferring one route over another are numerous. One approach for assigning OSPF link costs would be to set the costs to values inversely proportional to the bandwidth of the physical media. This would result in higher bandwidth routes having lower costs, thus becoming the preferred routes.

The cost values of RIP links are generally set to a value of 1. This results in the cost of a route to a destination being the number of hops to reach the destination. In the sample network, this would result in the three possible RIP routes from TCPCS7 to router 3.3.3.3 having the following costs:

- Direct (TCPCS7 -> 3.3.3.3), cost = 1
- Via TCPCS4 (TCPCS7 -> TCPCS4 -> 3.3.3.3), cost = 2

- Via router 8.8.8.8 and TCPCS4 (TCPCS7 -> 8.8.8.8 -> TCPCS4 -> TCPCS3), cost = 3

If it were desired that the route via TCPCS4 be the preferred route, this could be accomplished by increasing the cost of getting directly from TCPCS7 to router 3.3.3.3. This could be done by increasing either the OUT_METRIC configured on the RIP_INTERFACE statement for 9.67.102.3 on router 3.3.3.3 or the IN_METRIC configured on the RIP_INTERFACE statement for 9.67.102.7 on TCPCS7. Care must be taken when increasing IN_METRIC and OUT_METRIC values to be sure that the cost to reach any destination does not exceed the RIP maximum of 15.

Step 6: Configuring Virtual Links (If OSPF Protocol is Used)

The OSPF protocol is dependent upon complete connectivity of the backbone area. To maintain backbone connectivity each backbone router must be interconnected. If the configuration of an OSPF Autonomous System is such that the backbone area will become separated into two or more disconnected sections, connectivity must be restored for the protocol to work correctly. This can be done via a Virtual Link. An OSPF Virtual Link should not be confused with a VIPA link. Virtual Links can be configured between any two backbone routers that have an interface to a common non-backbone area. The VIRTUAL_LINK statements specify the ROUTERID of the link endpoint and must be configured at both endpoints. In the sample network shown in Figure 18 on page 146, a Virtual Link is configured between TCPCS4 and TCPCS7 to restore backbone connectivity through Area 1.1.1.1.

Sample TCPCS4

```
TCPCS4:
VIRTUAL_LINK
   Virtual_Endpoint_RouterID=7.7.7.7
Links_Transit_Area=1.1.1.1;
```

Sample TCPCS7

```
TCPCS7:
VIRTUAL_LINK
Virtual_Endpoint_RouterID=4.4.4.4
Links_Transit_Area=1.1.1.1;
```

Step 7: Managing High-Cost Links (If OSPF Protocol is Used)

The periodic nature of OSPF routing traffic requires a link's underlying data-link connection to be constantly open. This can result in unwanted usage charges on network segments whose costs are very high. There are two configuration steps that can be taken to inhibit the periodic nature of the protocol.

The first step that can be taken is to define the link as a Demand Circuit. The global Demand_Circuit=YES configuration statement must be specified before any links can be defined as demand circuits. If you configure an OSPF_INTERFACE with the Demand_Circuit=YES parameter, Link State Advertisements (LSAs) sent over the interface will not be periodically refreshed. Only LSAs with real changes will be readvertised. In addition, aging of these LSAs will be disabled such that they will not age out of the link state database.

Another step that can be taken is to define Hello Suppression for the link (using the Hello_Suppression parameter of the OSPF_INTERFACE statement). Hello Suppression is only meaningful if Demand_Circuit=YES and the device is point-to-point or point-to-multipoint. Refer to *z/OS Communications Server: IP Configuration Reference* for more information on configuring the Hello_Suppression parameter.

If Demand_Circuit=YES and Hello Suppression is implemented, the PP_Poll_Interval parameter of the OSPF_INTERFACE statement can be used to specify the interval at which OMPROUTE should attempt to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available.

Step 8: Defining Filters (If RIP Protocol is Used)

RIP Filters can be configured to OMPROUTE such that certain RIP routing information will not be broadcast out to other routers and/or accepted from other routers. The filters can be applied to individual RIP_INTERFACEs, via the FILTER parameter, or to all RIP interfaces via by the global FILTER statement. When defining a filter, a filter type (sending or receiving) is specified along with a destination/mask address pair. By using filters, an installation can limit the amount of RIP routing information broadcast into the network and/or the amount of RIP routing information maintained by OMPROUTE. In addition, filters can be used to hide destination addresses from portions of the network.

In the sample network shown in Figure 18 on page 146, if you wanted to hide the 10.1.1.0 subnet from TCPCS6 (as well as all routers and hosts on the remote side of TCPCS6), you could define the following filter on TCPCS7:

```
Filter=(nosend,10.1.1.0,255.255.255.0);
```

Step 9: Defining Route Precedence in a MultiProtocol Environment (If OSPF Protocol is Used)

Note that this discussion of route precedence is quite complicated. If OSPF is the only routing protocol used in your network, route precedence is less of a concern. If, in addition, none of your OSPF routers are configured as AS Boundary Routers, the route precedence concern is entirely eliminated. For environments with multiple protocols or AS Boundary Routers, the following information is provided.

OMPROUTE applies an order of precedence in choosing between two routes to the same destination that were learned via different routing protocols or using information provided by an OSPF AS Boundary Router. To describe this order of precedence applied by OMPROUTE, a few terms must first be defined.

RIP Route

A route learned via the RIP protocol. A RIP route is generated using information provided in a RIP packet from a neighboring router. For example, in the sample network shown in Figure 18 on page 146, the route from TCPCS7 to destination subnet 30.1.1.0 is a RIP route.

OSPF Internal Route

A route learned via the OSPF protocol where the entire path traversed to reach the destination lies within the OSPF autonomous system. For example, in the sample network shown in Figure 18 on page 146, the route from TCPCS7 to destination 9.67.108.2 on Router 2.2.2.2 is an OSPF internal route.

OSPF External Route

A route learned via the OSPF protocol where part of the path traversed to reach the destination does not lie within the OSPF autonomous system. The path will leave the autonomous system if it uses information brought into the OSPF autonomous system by an AS Boundary Router. This information brought into the OSPF AS may be information imported from a different autonomous system (for example, RIP) or information about destinations statically configured on or directly connected to the AS

Boundary Router. For example, in the sample network, shown in Figure 18 on page 146, the route from TCPCS4 to destination 9.67.103.6 on TCPCS6 is an OSPF external route. TCPCS7, configured as an AS Boundary Router, has imported information about that destination into the OSPF AS from the RIP AS.

OSPF external routes fall into two categories based upon the setting of the multiprotocol comparison value, which is defined in "MultiProtocol Comparison". If the comparison value is set to Type1 on the AS Boundary Router that imports the external information into the OSPF AS, then OSPF external routes generated using this information will be OSPF Type 1 External Routes. If the comparison value is set to Type2 on the AS Boundary Router, then the generated routes will be OSPF Type 2 External Routes. For example, in the sample network, shown in Figure 18 on page 146, if the comparison value on TCPCS7 (an AS Boundary Router) is set to Type 1, the route from TCPCS4 to destination 9.67.103.6 on TCPCS6 is an OSPF Type 1 external route. If the comparison value on TCPCS7 is set to Type 2, the route is an OSPF Type 2 external route.

MultiProtocol Comparison

You can configure this comparison value to allow for the specification of how route costs from different autonomous systems should be treated when they co-exist. In OMPROUTE, you can configure this value via the COMPARISON configuration statement. When COMPARISON=Type1 is configured, the route cost values used within different autonomous systems (for example, the OSPF AS and the RIP AS) are considered comparable. With COMPARISON=Type2 configured, the route cost values used with the different autonomous systems are considered non-comparable.

The comparison value can be used in several different ways, depending on the function being performed by a router:

- As an AS Boundary Router, OMPROUTE uses the comparison value to determine the type of external routes (Type 1 or Type 2) that is generated by routers in the OSPF AS using routing information that the AS Boundary Router imports into the OSPF AS. See "Step 9: Defining Route Precedence in a MultiProtocol Environment (If OSPF Protocol is Used)" on page 172 for additional OSPF external route definition information.
- As an AS Boundary Router, OMPROUTE also uses the comparison value in determining how route cost values will be assigned when importing routes from the OSPF AS into the RIP AS.
  - When COMPARISON=Type1 is configured (indicating that cost values are comparable), an OSPF route imported into the RIP AS will be advertised with the actual cost of the OSPF route. For example, in the sample network, if TCPCS7 is configured with COMPARISON=Type1 and the OSPF route from TCPCS7 to destination 9.67.108.2 on TCPCS2 has a cost of 7, then TCPCS7 will advertise into the RIP AS a RIP route to that destination with a cost of 7.

    Notes:

    1. An exception to this rule (defining how OSPF routes are advertised into the RIP AS when COMPARISON=Type1) occurs when the OSPF route to be imported is an OSPF Type 2 External Route. When this is the case, the route is not advertised into the RIP AS at all.
    2. It is important to remember the requirement that all destinations in the RIP AS must be reachable with a cost no greater than 15. Using COMPARISON=Type1 requires that the cost values of OSPF routes be

low. Any destinations in the OSPF AS that can only be reached from the RIP AS with a cost greater than 15 will become unreachable.

– When COMPARISON=Type2 is configured (indicating that cost values are non-comparable), an OSPF route imported into the RIP AS is advertised with a cost of 1. If a router in the RIP AS has two possible routes to a destination, one internal to the RIP AS and another that was imported from OSPF, this approach results in the route imported from OSPF being favored. For example, in the sample network, Figure 18 on page 146, if TCPCS7 is configured with COMPARISON=Type2 and TCPCS7 can somehow reach a destination in the 30.1.1.0 subnet without passing through TCPCS6 (using links not shown in the sample), then TCPCS7 advertises into the RIP AS a RIP route to the destination with a cost of 1. As a result, TCPCS6 determines that the destination can be reached via TCPCS7 with a cost of 2. If the cost of the route for TCPCS6 to reach the destination internal to the RIP AS is greater than 2, then the route via TCPCS7 is chosen.

  Note: An exception to this rule (defining how OSPF routes are advertised into the RIP AS when COMPARISON=Type2) occurs when the OSPF route to be imported is an OSPF Type 2 External Route. When this is the case, the route is advertised into the RIP AS with the actual cost of the OSPF Type 2 External Route.

- As any router that has routing information from different autonomous systems, OMPROUTE uses the comparison value while choosing between the routes generated using the information from the different autonomous systems. How the comparison value is used in this case is shown in table xxx.

Given these definitions, the order of precedence used in choosing between multiple routes to the same destination which were learned via the different protocols or by using information provided by an OSPF AS Boundary Router can be shown in Table 12. In Table 12, *Source Comparison* refers to the setting of the comparison value (using the COMPARISON configuration statement) on the router that is using the order of precedence to choose between the multiple routes, while *Route 1* and *Route 2* are the two possible routes being chosen between.

Table 12. Route Precedence

Source Comparison	Route 1 Type	Route 2 Type	Route Chosen
Type 1	OSPF Internal	RIP	OSPF Internal
Type 1	OSPF Internal	OSPF Type 1 External	OSPF Internal
Type 1	OSPF Internal	OSPF Type 2 External	OSPF Internal
Type 1	RIP	OSPF Type 1 External	Lowest Cost Route
Type 1	RIP	OSPF Type 2 External	RIP Route
Type 1	OSPF Type 1 External	OSPF Type 2 External	OSPF Type 1 External
Type 2	OSPF Internal	RIP	OSPF Internal
Type 2	OSPF Internal	OSPF Type 1 External	OSPF Internal
Type 2	OSPF Internal	OSPF Type 2 External	OSPF Internal

Table 12. Route Precedence (continued)

Source Comparison	Route 1 Type	Route 2 Type	Route Chosen
Type 2	RIP	OSPF Type 1 External	OSPF Type 1 External
Type 2	RIP	OSPF Type 2 External	Lowest Cost Route
Type 2	OSPF Type 1 External	OSPF Type 2 External	OSPF Type 1 External

Verification of OMPROUTE Configuration and State

The following sections show sample output from each of the commands that can be used to display OMPROUTE information. The syntax of these DISPLAY commands, as well as detailed information about the data displayed, can be found in *z/OS Communications Server: IP Configuration Reference*.

Displaying All OSPF Configuration Information

To display all of the OSPF configuration information, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,ALL
EZZ7831I GLOBAL CONFIGURATION 735
    TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
    STACK AFFINITY:          TCPCS7
    OSPF PROTOCOL:           ENABLED
    EXTERNAL COMPARISON:     TYPE 2
    AS BOUNDARY CAPABILITY:  ENABLED
    IMPORT EXTERNAL ROUTES:  RIP SUB
    ORIG. DEFAULT ROUTE:     NO
    DEFAULT ROUTE COST:      (1, TYPE 2)
    DEFAULT FORWARD. ADDR.:  0.0.0.0
    DEMAND CIRCUITS:         ENABLED

EZZ7832I AREA CONFIGURATION
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.0          0=NONE           NO          N/A           N/A
1.1.1.1          0=NONE           NO          N/A           N/A

--AREA RANGES--
AREA ID          ADDRESS          MASK             ADVERTISE?
1.1.1.1          9.67.101.0       255.255.255.0    NO

EZZ7833I INTERFACE CONFIGURATION
IP ADDRESS      AREA             COST  RTRNS  TRNSDLY PRI  HELLO   DEAD
7.7.7.7         1.1.1.1             1      5      1     1     10     40
9.67.104.7      1.1.1.1             1      5      1     1     10     40
9.67.100.7      1.1.1.1             1      5      1     1     10     40
9.67.102.7      1.1.1.1             1      5      1     1     10     40
9.67.106.7      1.1.1.1             1      5      1     1     10     40
9.67.107.7      0.0.0.0             1      5      1     1     10     40

EZZ7836I VIRTUAL LINK CONFIGURATION
VIRTUAL ENDPOINT     TRANSIT AREA      RTRNS  TRNSDLY HELLO  DEAD
4.4.4.4              1.1.1.1              10      5     30   180

EZZ7835I NBMA CONFIGURATION
                INTERFACE ADDR      POLL INTERVAL
                        9.67.104.7          180
EZZ7834I NEIGHBOR CONFIGURATION
                NEIGHBOR ADDR     INTERFACE ADDRESS   DR ELIGIBLE?
                9.67.104.15       9.67.104.7          YES
                9.67.104.25       9.67.104.7          NO
                9.67.104.16       9.67.104.7
```

Displaying Information about Configured OSPF Areas

To display information about configured OSPF Areas, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,AREAS
EZZ7832I AREA CONFIGURATION 737
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.0          0=NONE           NO          N/A           N/A
1.1.1.1          0=NONE           NO          N/A           N/A

--AREA RANGES--
AREA ID          ADDRESS          MASK             ADVERTISE?
1.1.1.1          9.67.101.0       255.255.255.0    NO
```

Displaying Information about Configured OSPF Interfaces

To display information about configured OSPF interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,IFS
EZZ7833I INTERFACE CONFIGURATION 739
IP ADDRESS      AREA             COST  RTRNS  TRNSDLY PRI  HELLO   DEAD
7.7.7.7         1.1.1.1             1      5      1     1     10     40
9.67.104.7      1.1.1.1             1      5      1     1     10     40
9.67.100.7      1.1.1.1             1      5      1     1     10     40
9.67.102.7      1.1.1.1             1      5      1     1     10     40
9.67.106.7      1.1.1.1             1      5      1     1     10     40
9.67.107.7      0.0.0.0             1      5      1     1     10     40
```

Displaying Information about Configured Non-broadcast Multiple Access OSPF Interfaces

To display information about configured Non-broadcast Multiple Access OSPF interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,NBMA
EZZ7835I NBMA CONFIGURATION 745
                INTERFACE ADDR      POLL INTERVAL
                9.67.104.7          180
```

Displaying Information about Configured OSPF Virtual Links

To display information about configured OSPF virtual links, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,VLINKS
EZZ7836I VIRTUAL LINK CONFIGURATION 747
VIRTUAL ENDPOINT     TRANSIT AREA      RTRNS  TRNSDLY HELLO  DEAD
4.4.4.4              1.1.1.1              10      5     30   180
```

Displaying Information about Configured OSPF Neighbors

To display information about configured OSPF neighbors enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,NBRS
EZZ7834I NEIGHBOR CONFIGURATION 749
                NEIGHBOR ADDR     INTERFACE ADDRESS   DR ELIGIBLE?
                9.67.104.15       9.67.104.7          YES
                9.67.104.25       9.67.104.7          NO
                9.67.104.16       9.67.104.7          NO
```

Displaying the Contents of a Single OSPF Link State Advertisement

To display the contents of a single OSPF link state advertisement, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LSA,LSTYPE=1,LSID=7.7.7.7,ORIG=7.7.7.7,AREAID=1.1.1.1
EZZ7880I LSA DETAILS 751
        LS AGE:          521
        LS OPTIONS:      E,DC
        LS TYPE:         1
        LS DESTINATION (ID): 7.7.7.7
```

```
LS ORIGINATOR:   7.7.7.7
LS SEQUENCE NO:  0X80000013
LS CHECKSUM:     0XA9A
LS LENGTH:       120
ROUTER TYPE:  ABR,ASBR,V
# ROUTER IFCS:   8
        LINK ID:          7.7.7.6
        LINK DATA:        255.255.255.254
        INTERFACE TYPE:   3
                NO. OF METRICS: 0
                TOS 0  METRIC:  1
        LINK ID:          8.8.8.8
        LINK DATA:        9.67.100.7
        INTERFACE TYPE:   1
                NO. OF METRICS: 0
                TOS 0 METRIC:   1 (1)
        LINK ID:          3.3.3.3
        LINK DATA:        9.67.102.7
        INTERFACE TYPE:   1
                NO. OF METRICS: 0
                TOS 0 METRIC:   1 (1)
        LINK ID:          4.4.4.4
        LINK DATA:        9.67.106.7
        INTERFACE TYPE:   1
                NO. OF METRICS: 0
                TOS 0 METRIC:   1 (1)
        LINK ID:          7.7.7.7
        LINK DATA:        255.255.255.255
        INTERFACE TYPE:   3
                NO. OF METRICS: 0
                TOS 0 METRIC:   1
        LINK ID:          9.67.100.8
        LINK DATA:        255.255.255.255
        INTERFACE TYPE:   3
                NO. OF METRICS: 0
                TOS 0 METRIC:   1
        LINK ID:          9.67.102.3
        LINK DATA:        255.255.255.255
        INTERFACE TYPE:   3
                NO. OF METRICS: 0
                TOS 0 METRIC:   1
        LINK ID:          9.67.106.4
        LINK DATA:        255.255.255.255
        INTERFACE TYPE:   3
                NO. OF METRICS: 0
                TOS 0 METRIC:   1
```

Displaying Statistics and Parameters for OSPF Areas

To display statistics and parameters for all OSPF areas attached to the router, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,AREASUM
EZZ7848I AREA SUMMARY 757
AREA ID        AUTHENTICATION   #IFCS  #NETS  #RTRS  #BRDRS DEMAND
0.0.0.0           NONE              2      0      4      2  ON
1.1.1.1           NONE              5      0      4      2  ON
```

Displaying the List of AS External Advertisements

To display a list of AS external advertisements that are in the OSPF link state database, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,EXTERNAL
EZZ7853I AREA LINK STATE DATABASE 759
TYPE LS DESTINATION     LS ORIGINATOR     SEQNO     AGE   XSUM
  5 @6.6.6.6            7.7.7.7         0X80000007  825  0X1B5C
  5 @9.67.103.6         7.7.7.7         0X80000007  831  0XE1F3
  5 @10.1.1.0           2.2.2.2         0X80000003 1690  0X2775
```

```
  5 @10.1.1.1           2.2.2.2         0X80000003 1690  0X1D7E
  5 @20.1.1.0           5.5.5.5         0X80000003 1616  0X4A3C
  5 @20.1.1.1           5.5.5.5         0X80000003 1616  0X4045
  5 @30.0.0.0           7.7.7.7         0X80000006  831  0XB0C0
  5 @30.1.1.0           7.7.7.7         0X80000006  831  0X99D5
  5 @30.1.1.4           7.7.7.7         0X80000001  825  0X7BF4
  5 @30.1.1.8           7.7.7.7         0X80000001  825  0X5319
  5 @130.200.0.0        3.3.3.3         0X80000003 1695  0X98C0
  5 @130.200.0.0        8.8.8.8         0X80000003 1630  0X243
  5 @130.200.1.1        3.3.3.3         0X80000003 1695  0X83D3
  5 @130.200.1.18       8.8.8.8         0X80000003 1630  0X42EF
  5 @130.201.0.0        3.3.3.3         0X80000003 1695  0X8CCB
  5 @130.201.0.0        8.8.8.8         0X80000003 1630  0XF54E
  5 @130.202.0.0        3.3.3.3         0X80000003 1694  0X80D6
  5 @130.202.0.0        8.8.8.8         0X80000003 1629  0XE959
               # ADVERTISEMENTS:       18
               CHECKSUM TOTAL:         0X83472
```

Displaying a List of Non-AS External Advertisements

To display a list of non-AS external advertisements that are in the OSPF link state database for a particular OSPF area, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,DATABASE,AREAID=1.1.1.1
EZZ7853I AREA LINK STATE DATABASE 761
TYPE LS DESTINATION     LS ORIGINATOR     SEQNO     AGE   XSUM
  1 @3.3.3.3            3.3.3.3         0X8000000F  879  0X8B11
  1 @4.4.4.4            4.4.4.4         0X8000001A  713  0XA020
  1 @7.7.7.7            7.7.7.7         0X80000013  711  0XA9A
  1 @8.8.8.8            8.8.8.8         0X8000000D  861  0XBD81
  3 @2.2.2.2            4.4.4.4         0X80000003 1676  0XC45C
  3 @5.5.5.4            7.7.7.7         0X80000003  880  0XE327
  3 @5.5.5.5            7.7.7.7         0X80000003  880  0XDF29
  3 @7.7.7.6            7.7.7.7         0X80000001  710  0X956E
  3 @9.67.107.5         7.7.7.7         0X80000006  881  0X4A14
  3 @9.67.107.7         7.7.7.7         0X80000003  880  0X4618
  3 @9.67.108.2         4.4.4.4         0X80000003 1667  0XBDB1
  3 @9.67.108.4         4.4.4.4         0X80000003 1658  0XB3B8
  4 @2.2.2.2            4.4.4.4         0X80000003 1658  0XAC74
  4 @5.5.5.5            7.7.7.7         0X80000003  880  0XC741
               # ADVERTISEMENTS:       14
               CHECKSUM TOTAL:         0X884B0
```

Displaying Current, Run-time Statistics and Parameters for OSPF Interfaces

To display current, run-time statistics and parameters for OSPF interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,INTERFACE
EZZ7849I INTERFACES 763
IFC ADDRESS     PHYS         ASSOC. AREA     TYPE   STATE  #NBRS  #ADJS
7.7.7.7         VIPA1A       1.1.1.1         VIPA    N/A    N/A    N/A
9.67.104.7      NBMA7        1.1.1.1         MULTI     1      3      0
9.67.100.7      CTC7TO8      1.1.1.1         P-P      16      1      1
9.67.102.7      CTC7TO3      1.1.1.1         P-P      16      1      1
9.67.106.7      CTC7TO4      1.1.1.1         P-P      16      1      1
9.67.107.7      CTC7TO5      0.0.0.0         P-P      16      1      1
UNNUMBERED      VL/0         0.0.0.0         VLINK    16      1      1
```

Displaying Current, Run-time Statistics and Parameters for a Specific OSPF Interface

To display current, run-time statistics and parameters for a specific OSPF interface, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,IF,NAME=CTC7TO4
EZZ7850I INTERFACE DETAILS 769
               INTERFACE ADDRESS:      9.67.106.7
```

```
                ATTACHED AREA:          1.1.1.1
                PHYSICAL INTERFACE:     CTC7TO4
                INTERFACE MASK:         255.255.255.0
                INTERFACE TYPE:         P-P
                STATE:                  16
                DESIGNATED ROUTER:      0.0.0.0
                BACKUP DR:              0.0.0.0

  DR PRIORITY:       1  HELLO INTERVAL:   10  RXMT INTERVAL:     5
  DEAD INTERVAL:    40  TX DELAY:          1  POLL INTERVAL:     0
  DEMAND CIRCUIT:  OFF  HELLO SUPPRESS:  OFF  SUPPRESS REQ:    OFF
  MAX PKT SIZE:   1024  TOS 0 COST:        1

  # NEIGHBORS:       1  # ADJACENCIES:     1  # FULL ADJS.:      1
  # MCAST FLOODS:   15  # MCAST ACKS:      4  DL UNICAST:      OFF
  MC FORWARDING:   OFF

  NETWORK CAPABILITIES:
   POINT-TO-POINT
   DEMAND-CIRCUITS
```

Displaying Current, Run-time Statistics and Parameters for OSPF Neighbors

To display current, run-time statistics and parameters for OSPF neighbors, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,NBR
EZZ7851I NEIGHBOR SUMMARY 771
NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
9.67.104.16     0.0.0.0             1     0     0     0  OFF NBMA7
9.67.104.25     0.0.0.0             1     0     0     0  OFF NBMA7
9.67.104.15     0.0.0.0             1     0     0     0  OFF NBMA7
9.67.100.8      8.8.8.8           128     0     0     0  OFF CTC7TO8
9.67.102.3      3.3.3.3           128     0     0     0  OFF CTC7TO3
9.67.106.4      4.4.4.4           128     0     0     0  OFF CTC7TO4
9.67.107.5      5.5.5.5           128     0     0     0  OFF CTC7TO5
VL/0            4.4.4.4           128     0     0     0  OFF *
```

Displaying Current Run-time Statistics and Parameters for a Specific OSPF Neighbor

To display current run-time statistics and parameters for a specific OSPF neighbor, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,NBR,IPADDR=9.67.106.4
EZZ7852I NEIGHBOR DETAILS 779
                NEIGHBOR IP ADDRESS:    9.67.106.4
                OSPF ROUTER ID:         4.4.4.4
                NEIGHBOR STATE:         128
                PHYSICAL INTERFACE:     CTC7TO4
                DR CHOICE:              0.0.0.0
                BACKUP CHOICE:          0.0.0.0
                DR PRIORITY:            1
                NBR OPTIONS:            E
  DB SUMM QLEN:      0  LS RXMT QLEN:      0  LS REQ QLEN:       0
  LAST HELLO:        4  NO HELLO:        OFF
  # LS RXMITS:       1  # DIRECT ACKS:     0  # DUP LS RCVD:     6
  # OLD LS RCVD:     0  # DUP ACKS RCVD:   1  # NBR LOSSES:      0
```

Displaying Routes to Other Routers that have been Calculated by OSPF

To display routes to other routers that have been calculated by OSPF, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,ROUTERS
EZZ7855I OSPF ROUTERS 781
DTYPE RTYPE DESTINATION       AREA           COST       NEXT HOP(S)
ASBR  SPF   2.2.2.2          0.0.0.0          2        9.67.106.4
```

```
  BR  SPF   4.4.4.4          0.0.0.0          1         9.67.106.4
ASBR  SPF   5.5.5.5          0.0.0.0          1         9.67.107.5
ASBR  SPF   3.3.3.3          1.1.1.1          1         9.67.102.3
  BR  SPF   4.4.4.4          1.1.1.1          1         9.67.106.4
ASBR  SPF   8.8.8.8          1.1.1.1          1         9.67.100.8
```

Displaying the Number of LSAs Currently in the Link State Database

To display the number of LSAs currently in the link state database, categorized by type, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,DBSIZE
EZZ7854I LINK STATE DATABASE SIZE 783
                # ROUTER-LSAS:            8
                # NETWORK-LSAS:           0
                # SUMMARY-LSAS:           37
                # SUMMARY ROUTER-LSAS:    7
                # AS EXTERNAL-LSAS:       18
                # INTRA-AREA ROUTES:      24
                # INTER-AREA ROUTES:      1
                # TYPE 1 EXTERNAL ROUTES: 0
```

Displaying Statistics Generated by the OSPF Routing Protocol

To display statistics generated by the OSPF routing protocol, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,STATS
EZZ7856I OSPF STATISTICS 785
                OSPF ROUTER ID:         7.7.7.7
                EXTERNAL COMPARISON:    TYPE 2
                AS BOUNDARY CAPABILITY: YES
                IMPORT EXTERNAL ROUTES: RIP SUB
                ORIG. DEFAULT ROUTE:    NO
                DEFAULT ROUTE COST:     (1, TYPE 2)
                DEFAULT FORWARD. ADDR.: 0.0.0.0
ATTACHED AREAS:                  2  OSPF PACKETS RCVD:              821
OSPF PACKETS RCVD W/ERRS:        0  TRANSIT NODES ALLOCATED:         55
TRANSIT NODES FREED:            47  LS ADV. ALLOCATED:              263
LS ADV. FREED:                 201  QUEUE HEADERS ALLOC:             96
QUEUE HEADERS AVAIL:            96  MAXIMUM LSA SIZE:               976
# DIJKSTRA RUNS:                9  INCREMENTAL SUMM. UPDATES:        4
INCREMENTAL VL UPDATES:          0  MULTICAST PKTS SENT:            746
UNICAST PKTS SENT:             107  LS ADV. AGED OUT:                 0
LS ADV. FLUSHED:                22  PTRS TO INVALID LS ADV:           0
INCREMENTAL EXT. UPDATES:       49
```

Displaying the Routes in the OMPROUTE Routing Table

To display all of the routes in the OMPROUTE routing table, enter the following command:

```
D TCPIP,TCPCS7,OMP,RTTABLE
EZZ7847I ROUTING TABLE 796
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SBNT   2.0.0.0          FF000000  1       1368    NONE
 SPF   2.2.2.2          FFFFFFFE  3       1380    9.67.106.4
 SPF   2.2.2.2          FFFFFFFF  3       1380    9.67.106.4
SBNT   3.0.0.0          FF000000  1       1549    NONE
 SPF   3.3.3.2          FFFFFFFE  2       1561    9.67.102.3
 SPF   3.3.3.3          FFFFFFFF  2       1561    9.67.102.3
SBNT   4.0.0.0          FF000000  1       1549    NONE
 SPF   4.4.4.4          FFFFFFFE  2       1561    9.67.106.4
 SPF   4.4.4.4          FFFFFFFF  2       1561    9.67.106.4
SBNT   5.0.0.0          FF000000  1       1549    NONE
 SPF   5.5.5.4          FFFFFFFE  2       1567    9.67.107.5
 SPF   5.5.5.5          FFFFFFFF  2       1567    9.67.107.5
SBNT   6.0.0.0          FF000000  1       1549    NONE
```

```
 RIP   6.6.6.6          FFFFFFFE  2       30      9.67.103.6
SBNT   7.0.0.0          FF000000  1       1368    NONE
SPIA*  7.7.7.6          FFFFFFFE  3       1380    9.67.106.4
 DIR*  7.7.7.7          FFFFFFFF  1       1574    VIPA1A
SBNT   8.0.0.0          FF000000  1       1549    NONE
 SPF   8.8.8.8          FFFFFFFE  2       1545    9.67.100.8
 SPF   8.8.8.8          FFFFFFFF  2       1545    9.67.100.8
SBNT   9.0.0.0          FF000000  1       1368    NONE
 DIR*  9.67.100.0       FFFFFF00  1       1576    9.67.100.7
 SPF   9.67.100.7       FFFFFFFF  2       1545    CTC7TO8
 SPF   9.67.100.8       FFFFFFFF  1       1572    9.67.100.8
 SPF   9.67.101.3       FFFFFFFF  2       1561    9.67.106.4
 SPF   9.67.101.4       FFFFFFFF  2       1561    9.67.102.3
 DIR*  9.67.102.0       FFFFFF00  1       1575    9.67.102.7
 SPF   9.67.102.3       FFFFFFFF  1       1566    9.67.102.3
 SPF   9.67.102.7       FFFFFFFF  2       1561    CTC7TO3
 DIR*  9.67.103.0       FFFFFF00  1       1575    9.67.103.7
 RIP   9.67.103.6       FFFFFFFF  1       30      9.67.103.6
 SPF   9.67.105.4       FFFFFFFF  2       1545    9.67.100.8
 SPF   9.67.105.8       FFFFFFFF  2       1561    9.67.106.4
 DIR*  9.67.106.0       FFFFFF00  1       1576    9.67.106.7
 SPF   9.67.106.4       FFFFFFFF  1       1566    9.67.106.4
 SPF   9.67.106.7       FFFFFFFF  2       1561    CTC7TO4
 DIR*  9.67.107.0       FFFFFF00  1       1577    9.67.107.7
 SPF   9.67.107.5       FFFFFFFF  1       1574    9.67.107.5
 SPF   9.67.107.7       FFFFFFFF  2       1566    CTC7TO5
 SPF   9.67.108.2       FFFFFFFF  2       1380    9.67.106.4
 SPF   9.67.108.4       FFFFFFFF  3       1380    9.67.106.4
SBNT   10.0.0.0         FF000000  1       1368    NONE
SPE2   10.1.1.0         FFFFFF00  0       1379    9.67.106.4
SPE2   10.1.1.1         FFFFFFFF  0       1379    9.67.106.4
SBNT   20.0.0.0         FF000000  1       1549    NONE
SPE2   20.1.1.0         FFFFFF00  0       1379    9.67.107.5
SPE2   20.1.1.1         FFFFFFFF  0       1379    9.67.107.5
 RIP   30.0.0.0         FF000000  2       30      9.67.103.6
 RIP   30.1.1.0         FFFFFF00  2       30      9.67.103.6
 RIP % 30.1.1.4         FFFFFFFF  2       30      9.67.103.6
 RIP % 30.1.1.8         FFFFFFFF  2       30      9.67.103.6
SPE2   130.200.0.0      FFFF0000  0       1379    9.67.100.8      (2)
SPE2   130.200.1.1      FFFFFFFF  0       1379    9.67.102.3
SPE2   130.200.1.18     FFFFFFFF  0       1379    9.67.100.8
SPE2   130.201.0.0      FFFF0000  0       1379    9.67.100.8      (2)
SPE2   130.202.0.0      FFFF0000  0       1379    9.67.100.8      (2)
                      0 NETS DELETED, 4 NETS INACTIVE
```

Displaying the Routes to a Specific Destination

To display information about the routes to a specific destination, enter the following command:

```
D TCPIP,TCPCS7,OMP,RTTABLE,DEST=130.201.0.0
EZZ7874I ROUTE EXPANSION 798
DESTINATION:    130.201.0.0
MASK:           255.255.0.0
ROUTE TYPE:     SPE2
DISTANCE:       0
AGE:            1485
NEXT HOP(S):    9.67.100.8        (CTC7TO8)
                9.67.102.3        (CTC7TO3)
```

Displaying All of the RIP Configuration Information

To display all of the RIP configuration information, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,LIST,ALL
EZZ7843I RIP CONFIGURATION 800
TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
```

```
STACK AFFINITY: TCPCS7
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
CTC7TO6          9.67.103.7      RIP-2 MULTICAST.
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0

EZZ7844I RIP ROUTE ACCEPTANCE
ACCEPT RIP UPDATES ALWAYS FOR:
  30.1.1.8          30.1.1.4
```

Displaying Information about Configured RIP Interfaces

To display information about configured RIP interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,LIST,IFS
EZZ7843I RIP CONFIGURATION 806
TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
STACK AFFINITY: TCPCS7
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
CTC7TO6          9.67.103.7      RIP-2 MULTICAST.
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0
```

Displaying the Routes to be Unconditionally Accepted

To display the routes to be unconditionally accepted, as configured with the Accept_RIP_Route statement, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,LIST,ACCEPTED
EZZ7844I RIP ROUTE ACCEPTANCE 808
ACCEPT RIP UPDATES ALWAYS FOR:
  30.1.1.8          30.1.1.4
```

Displaying Current Run-time Information about RIP Interfaces

To display current, run-time information about RIP interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,IF
EZZ7859I RIP INTERFACES 810
IFC ADDRESS      IFC NAME         SUBNET MASK     MTU   DESTINATION
9.67.103.7       CTC7TO6          255.255.255.0   1024  0.0.0.0
```

Displaying Current Run-time Information about a Specific RIP Interface

To display current, run-time information about a specific RIP interface, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,IF,NAME=CTC7TO6
EZZ7860I RIP INTERFACE DETAILS 812
INTERFACE ADDRESS:      9.67.103.7
INTERFACE NAME:         CTC7TO6
SUBNET MASK:            255.255.255.0
MTU                     1024
DESTINATION ADDRESS:    0.0.0.0

RIP VERSION:            2     SEND POIS. REV. ROUTES: YES
IN METRIC:              1     OUT METRIC:             0
RECEIVE NET ROUTES:     YES   RECEIVE SUBNET ROUTES:  YES
RECEIVE HOST ROUTES:    NO    SEND DEFAULT ROUTES:    NO
```

```
SEND NET ROUTES:        YES   SEND SUBNET ROUTES:     YES
SEND STATIC ROUTES:     NO    SEND HOST ROUTES:       NO

SEND ONLY: ALL
```

Displaying the Global RIP Filters

To display the global RIP filters, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,FILTERS
EZZ8016I GLOBAL RIP FILTERS 814
SEND ONLY: ALL

FILTERS: NOSEND          10.1.1.0            255.255.255.0
```

Sample OMPROUTE Configuration Files

The following is an example of a pure OSPF environment (from TCPCS4 in the Figure 18 on page 146).

```
RouterID=4.4.4.4;
Area
     Area_Number = 0.0.0.0;
Area
     Area_Number = 1.1.1.1;
OSPF_Interface
     IP_Address=9.67.108.4
     Name = CTC4TO2
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=0.0.0.0
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.106.4
     Name = CTC4TO7
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.105.4
     Name = CTC4TO8
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.101.4
     Name = CTC4TO3
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=4.4.4.4
     Name = VIPA1A
     Subnet_Mask=255.255.255.254
     Attaches_To_Area=1.1.1.1
     Cost0 = 1;
Virtual_Link
     Virtual_Endpoint_RouterID=7.7.7.7
     Links_Transit_Area=1.1.1.1;
```

The following is an example of mixed OSPF and RIP environments (from TCPCS7 in Figure 18 on page 146).

```
;*********************************
;  OSPF Configuration Statements  *
;*********************************
RouterID=7.7.7.7;
Area
     Area_Number = 0.0.0.0;
Area
     Area_Number = 1.1.1.1;
AS_Boundary_Routing
     Import_Subnet_Routes=YES
     Import_RIP_Routes=YES;
OSPF_Interface
     IP_Address=9.67.107.7
     Name = CTC7TO5
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=0.0.0.0
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.106.7
     Name = CTC7TO4
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.102.7
     Name = CTC7TO3
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.100.7
     Name = CTC7TO8
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.104.7
     Name = NBMA7
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     Non_Broadcast=YES
     NB_Poll_Interval=180
     MTU = 1024
     Cost0 = 1
     DR_Neighbor=9.67.104.15
     No_DR_Neighbor=9.67.104.16
     No_DR_Neighbor=9.67.104.25;
OSPF_Interface
     IP_Address=7.7.7.7
     Name = VIPA1A
     Subnet_Mask=255.255.255.254
     Attaches_To_Area=1.1.1.1
     Cost0 = 1;
Range
     IP_Address=9.67.101.0
     Subnet_Mask=255.255.255.0
     Area_Number=1.1.1.1
     Advertise=NO;
Virtual_Link
     Virtual_Endpoint_RouterID=4.4.4.4
     Links_Transit_Area=1.1.1.1;
;*********************************
;  RIP Configuration Statements  *
```

```
;*******************************
Originate_RIP_Default
     Condition=Always;
Accept_RIP_Route
     IP_Address=30.1.1.4;
Accept_RIP_Route
     IP_Address=30.1.1.8;
Filter=(nosend,10.1.1.0,255.255.255.0);
RIP_Interface
     IP_Address=9.67.103.7
     Name = CTC7TO6
     Subnet_Mask=255.255.255.0
     Receive_Dynamic_Hosts=NO
     MTU = 1024
     RipV2=YES;
```

The following is an example of a pure RIP environment (from TCPCS6 in Figure 18 on page 146).

```
RIP_Interface
     IP_Address=9.67.103.6
     Name = CTC6TO7
     Subnet_Mask=255.255.255.0
     MTU = 1024
     Send_Static_Routes=YES
     Send_Host_Routes=YES
     RipV2=YES;
Interface
     IP_Address=6.6.6.6
     Name = VIPA1A
     Subnet_Mask=255.255.255.254;
```

Verification of Routing (Static and Dynamic)

- If static routes are used, an indirect route must not be defined before the route to its first hop is defined. The following example shows an incorrect configuration.

  ```
  BEGINRoutes      ; first BEGINRoutes in the profile
  ;       Network/mask    FirstHop    LinkName PacketSize
    Route 9.67.104.0/24   9.67.105.8  CTC4TO8  MTU 1500
    Route 9.67.105.0/24   =           CTC4TO8  MTU 1500
  ENDRoutes
  ```

  When configured incorrectly, the following error message is displayed:

  ```
  EZZ0657I ROUTE LIST ENTRY NUMBER 1 ON LINE 28 FOR DESTINATION
  9.67.104.0 IS UNREACHABLE THROUGH INTERFACE 9.67.105.8 ON CTC4TO8
  ```

- If OMPROUTE is used for the OSPF protocol only and AUTOLOG is not configured correctly (see “Autolog Considerations for OMPROUTE” on page 157), OMPROUTE will be periodically restarted and the following messages are dislayed:

  ```
  $HASP100 OMPROUTE ON STCINRDR
  $HASP373 OMPROUTE STARTED
  IEF403I OMPROUT1 - STARTED
        OMPROUT1    OMPROUTE    BPXBATCH    0000
  EZZ7800I OMPROUTE STARTING
  EZZ7872I OMPROUTE FOUND ANOTHER ROUTING APPLICATION ALREADY ACTIVE
  EZZ8074I OMPROUTE PROCESSING ERROR
  EZZ7805I OMPROUTE EXITING ABNORMALLY - RC(11)
  OMPROUT1    *OMVSEX     BPXPRECP    0011
  IEF404I OMPROUT1 - ENDED
  $HASP395 OMPROUT1 ENDED
  ```

- If a configuration statement in the OMPROUTE configuration file has a missing semicolon, the syntax checker might issue the following message:

```
EZZ7830I SYNTAX ERROR AT LINE 22 OF OMPROUTE CONFIGURATION FILE
PROCESSING END OF FILE
```

Verifying Connections with NETSTAT, PING, and TRACERTE

The interfaces were verified with the instructions in “Chapter 1. Configuration Overview” on page 3. The first thing to verify is that the devices are started. In the case of point-to-point links like the CTCs in TCPCS4, the following message is written to the z/OS console when the device starts:

```
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE CTCE02
```

The same information can be determined from NETSTAT DEV. Following is a portion of the output of NETSTAT DEV with the CTCE02 device shown as ready. The NETSTAT DEV can be issued on TCPCS4 and TCPCS7 to verify that the devices on both systems are ready.

```
netstat dev
EZZ2350I MVS TCP/IP NETSTAT CS V2R10       TCPIP NAME: TCPCS4        02:56:25
 ......
EZZ2760I DevName: CTCE00            DevType: CTC       DevNum: 0E00
EZZ2766I   DevStatus: Ready
EZZ2761I   LnkName: CTC4TO7           LnkType: CTC         LnkStatus: Ready

EZZ2762I     NetNum: 0   QueSize: 0   ByteIn: 0000000000   ByteOut: 0000000000
EZZ2768I   BSD Routing Parameters:
EZZ2769I     MTU Size: 00000             Metric: 00
EZZ2770I     DestAddr: 0.0.0.0           SubnetMask: 255.255.255.0
EZZ2810I   Multicast Specific:
EZZ2811I     Multicast Capability: Yes
EZZ2812I     Group             RefCnt
EZZ2813I     -----             ------
EZZ2814I     224.0.0.1         0000000001
```

If the devices do not have a LnkStatus of Ready, this must be resolved before continuing. There are several things that might cause the LnkStatus to not be ready. For example, the device might be defined to z/OS incorrectly, the device might not be defined in PROFILE.TCPIP correctly, and so on.

You can PING each others hosts within the network to verify indirect routes exist. PING 9.67.107.7 from TCPCS4 to verify an indirect route.

```
ping 9.67.107.7
 EZA0458I Ping CS V2R10: Pinging host 9.67.107.7.  Use ATTN to interrupt.
 EZA0463I PING: Ping #1 response took 0.048 seconds. Successes so far 1.
 READY
```

To use TRACERTE to verify that the correct route is being taken for each first hop or adjcent router, use this command:

```
tracerte 7.7.7.7
 EZA0484I Trace route to 7.7.7.7 (7.7.7.7)
 EZA0505I 1   (7.7.7.7) 11 ms 14 ms 15 ms
 EZA0516I
 READY
```

To use TRACERTE to verify that the correct route is being taken for each a indirectly attached host, use this command:

```
tracerte 9.67.107.5
 EZA0484I Trace route to 9.67.107.5 (9.67.107.5)
 EZA0505I 1   (7.7.7.7) 18 ms 12 ms 17 ms
 EZA0505I 2   (9.67.107.5) 14 ms 17 ms 24 ms
 EZA0516I
 READY
```

Part 2. Server Applications

Chapter 5. Network Connectivity with a SNA Network

The objective of this chapter is to guide you through the steps required to implement:

- SNALINK LU0
- SNALINK LU6.2
- X.25 NPSI
- NCPROUTE

Before You Configure...

Read and understand “Chapter 1. Configuration Overview” on page 3. It covers important information about data set naming and search sequences.

SNALINK LU0 Environment

SNALINK allows TCP/IP to send and receive packets using SNA sessions instead of dedicating physical network hardware (such as a channel-to-channel adapter or channel connection to a 3745/46 Communication Controller).

Prior to NCP V7R3, NCP did not support cross-channel native IP transmission of the transport PDUs associated with RIP traffic. NCP expects these PDUs to be carried in SNA frames. SNALINK is therefore still required for installations where dynamic routing is performed with the NCP (via NCPROUTE). See *z/OS Communications Server: IP Configuration Reference* for more information.

SNALINK allows an installation to multiplex SNA and IP traffic over the same I/O subchannels, rather than requiring separate subchannels dedicated to VTAM and TCP/IP. While such multiplexing capability may be desirable at some installations, the native TCP/IP CTC and 3745/46 device drivers will likely outperform SNALINK connections. Interaction with the SNALINK address space is very CPU-intensive, and is not required with the native TCP/IP CTC and 3745/46 device drivers. (See the *z/OS Communications Server: IP Configuration Reference* for configuration information.) It is therefore important to weigh the multiplexing capability that SNALINK provides against its performance cost, in determining whether to use SNALINK or the native TCP/IP CTC or 3745/46 device drivers.

Understanding the SNALINK Environment

The SNALINK environment interfaces between the TCP/IP environment’s SNAIUCV driver and the customer’s SNA network. SNALINK communicates with one or more instances of SNALINK at remote nodes, using the SNA LU type 0 protocol. See Figure 19 on page 190 for a description of the SNALINK environment interfaces.

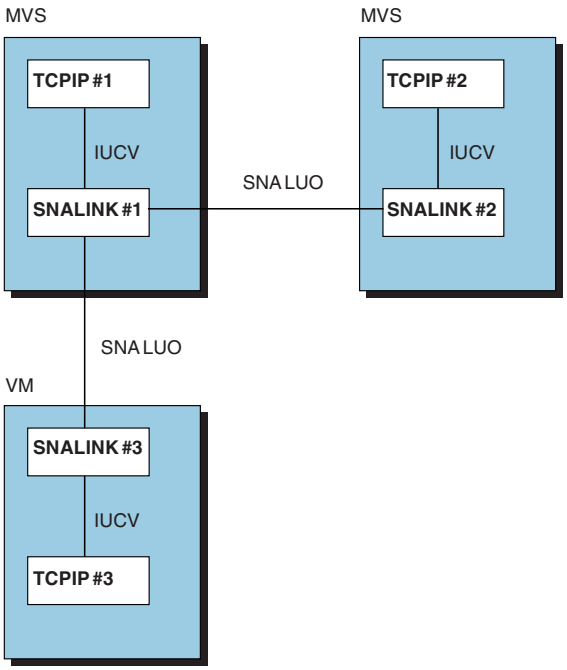


Figure 19. SNALINK Environment Interfaces

Each SNALINK environment can communicate with up to 9999 SNALINKs simultaneously. The number of connections is determined by the parameters you pass to the SNALINK cataloged procedure. The default is 6 sessions running in dual mode for a total of 3 SNALINKs.

- When operating in single mode, SNALINK opens one full duplex session.
- When operating in dual mode, SNALINK opens two System Network Architecture (SNA) sessions for each remote logical unit (LU) with which it communicates, one for sending and one for receiving.

Configuring SNALINK LU0

Steps to configure SNALINK LU0:

1. Specify configuration statements in *hlq*.PROFILE.TCPIP.
2. Update the SNALINK cataloged procedure.
3. Define the SNALINK application to VTAM.

Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP

The following sections describe the changes you must make to your TCPIP address space configuration data set (*hlq*.PROFILE.TCPIP).

Defining SNA DLC Links: SNA DLC links are point-to-point and require DEVICE and LINK statements in the configuration data set. The DLC link constitutes a separate network, even though it includes only two hosts. To define a link, each host to which the DLC link is attached requires:

- A pair of SNA LU0 DEVICE and LINK statements
- A HOME statement
- A BSDROUTINGPARMS statement or a GATEWAY or BEGINROUTES statement

SNA DLC links are defined in one of two ways:

- By unique network or subnetwork numbers, if the hosts to which they connect are not attached to other networks.

- By the IP address of the hosts to which they connect, if the hosts are attached to other networks.

You usually have to assign a unique network or subnetwork number to the SNALINK. If the link connects 2 hosts that also have other networks attached to them, the DLC link does not need its own subnetwork number. Figure 20 illustrates how to define an SNA DLC link if the 2 hosts are connected to other networks in the following way:

- Host A and Host B are connected by SNA DLC
- Host A is also connected to a token ring, 193.1.1
- Host B is also connected to a token ring, 193.1.2
- Host A's home address on its token ring is 193.1.1.1
- Host B's home address on its token ring is 193.1.2.1

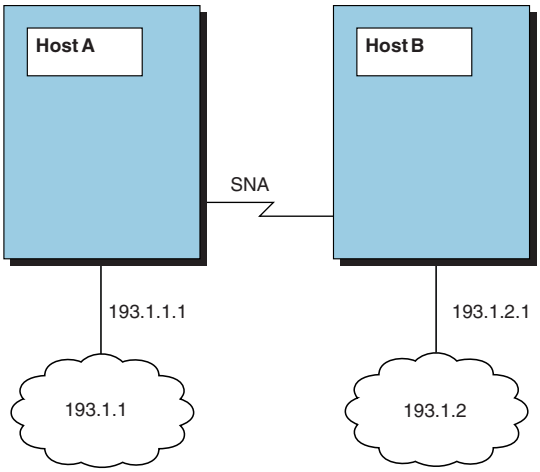


Figure 20. SNA DLC Link

Host A's *hlq*.PROFILE.TCPIP could contain:

```
DEVICE  LCS1 LCS BA0
LINK  TR1  IBMTR 0 LCS1
DEVICE SNALU0 SNAIUCV SNALINK LU000000 SNALINKA
LINK SNAIUCV1 SAMEHOST  1 SNALU0
HOME
    193.1.1.1  TR1
    193.1.1.2  SNAIUCV1

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    193.1.1.0      =          TR1         2000        0
    193.1.2.0      =          SNAIUCV1    2000        0
```

Host B's *hlq*.PROFILE.TCPIP could contain:

```
DEVICE LCS2 LCS BE0
LINK  TR1 IBMTR 0 LCS2
DEVICE SNALU0 SNAIUCV SNALINK LU000001 SNALINKA
LINK SNAIUCV1 SAMEHOST  1 SNALU0
HOME
    193.1.2.1  TR1
    193.1.2.2  SNAIUCV1

GATEWAY
```

```
;  Network         First hop  Link    Packet size    Subnet mask
    193.1.2.0      =          TR1         2000        0
    193.1.1.0      =          SNAIUCV1    2000        0
```

Notes:

1. The *lu_name* must be different on each host. In the example, the *lu_name* for Host A is lu000000. The *lu_name* for Host B is lu000001.
2. In the example, the *lu_name* for Host A is the remote or partner LU.

Hosts A and B are addressed by their token-ring home addresses, even if the packets reach them through the SNA DLC link.

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

Note: If you plan to run a network-monitoring protocol that requires each subnet to have its own subnet number, you can assign a separate subnet network number to the DLC link.

Defining NCPROUTE and 3745 LAN Attachments: If your TCP/IP configuration supports NCPROUTE or 3745 Communications Controller Ethernet or token-ring links, you must do the following:

- Match the *lu_name* on the DEVICE statement to the LU statement in NCST section of your NCP generation.

  The following example shows the LU name A04TOLU1 defined in the *hlq*.PROFILE.TCPIP DEVICE statements and in the NCP generation.

```
DEVICE  SNA1LINK SNAIUCV SNALINK A04TOLU1 SNAL1STC
LINK  SNALINK SAMEHOST 1 SNA1LINK

HOME
    9.67.116.66  SNALINK

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    9.67.116.65   =          SNALINK     2000        HOST

START SNA1LINK
```

```
*************************************************************************
*               NCST IP INTERFACES**
*************************************************************************
A04NCSTG GROUP NCST=IP,LNCTL=SDLC,VIRTUAL=YES
A04NCSTL LINE LINEFVT=CXSXFVT,PUFVT=CXSXFVT,LUFVT=(CXSXFVT,CXSXFVT),LIN*
              ECB=CXSXLNK
A04NCSTP PU VPACING=0,PUTYPE=2,PUCB=CXSP0000S
*
A04TOLU1 LU INTFACE=(NCSTALU1,1492),REMLU=SNALKLU1,LUCB=(CXSXL0000,CXSS0*
              000),LOCADDR=1
*************************************************************************
```

- Match the remote LU name SNALKLU1 in the NCP generation to the APPLID in the SNALINK cataloged procedure parameters and in the VTAM APPL definition.

```
//SNALINK  PROC MODULE=SNALINK,TCPID='TCPV3',APPLID='SNALKLU1'
//SNALINK  EXEC PGM=&MODULE<REGION=$4096K,TIME=1440,
           PARM='&TCPID &APPLID C7 6 0003 SINGLE'
```

For additional information on configuring these links, see *z/OS Communications Server: IP Configuration Reference*.

Step 2: Update the SNALINK Cataloged Procedure

Update the SNALINK cataloged procedure by copying the sample in SEZAINST(SNALPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify SNALINK parameters and change the DD statements, as required. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the SNALINK cataloged procedure.

Step 3: Define the SNALINK Application to VTAM

In dual mode, SNALINK opens 2 SNA sessions for each remote logical unit with which it communicates: one for sending and one for receiving. In single mode, SNALINK opens one full-duplex session.

Figure 21 is an example of a typical VTAM APPL statement for SNALINK. The application identifier (SNALKB03 in this example) must match the APPLID specified in the SNALINK cataloged procedure parameters.

```
SNALKB03  APPL ACBNAME=SNALKB03,                                X
               AUTH=(ACQ,VPACE),                                X
               SRBEXIT=YES,                                     X
               EAS=12,                                          X
               PARSESS=YES,                                     X
               SONSCIP=YES,                                     X
               VPACING=0
```

Figure 21. APPL Statement for SNALINK

Note: ***SRBEXIT must be YES.***

VTAM Considerations:

- Each connection requires 100KB of virtual storage.
- SNALINK provides its own BIND parameters, so it does not assume or require any particular LOGMODE entries.
- The EAS value should be two times the number of maximum sessions passed to the SNALINK cataloged procedure.
- SRBEXIT=YES.
- You might have to specify pacing values (VPACING). Consult your VTAM network administrator for further details.
- For *max_ru_size*, be sure to consider the size of the TH, RH, and RU portions. If the maximum size PIU exceeds MAXRU, the NCP issues a negative response with sense 800A0000 (PIU too long). The definition used in NCP and SNALINK must be such that MAXRU is at least 29 bytes less than MAXDATA. Refer to *z/OS Communications Server: SNA Network Implementation Guide* for more information on defining the MAXDATA, MAXBFRU, and UNITSZ operands.

Stopping and Starting SNALINK

If necessary, you can immediately retry a session that is waiting for the retry delay to expire by stopping and starting the SNALINK LU0 interface.

To stop SNALINK and close all connections, use the STOP command on the operator's console. For example, if SNALPROC was the name of the member in the cataloged procedure used to start SNALINK, you enter:

```
STOP SNALPROC
```

You can also stop SNALINK with the HALT parameter on the MODIFY command. See “Controlling the SNALINK LU0 Interface with the MODIFY Command” on page 196.

SNALINK can be started by:

- Restarting the TCPIP address space if you have included the SNALINK procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.
- Issuing START *procname* at the command console (where *procname* is the name of the member in the cataloged procedure used to start the SNALINK LU0 interface).

For example, to restart SNALPROC, enter

```
START SNALPROC
```

Sample Console

The example in Figure 22 and the accompanying information illustrate SNALINK operation.

The line number notations in the example have been added for clarity. They do not appear in the console output.

```
Line 1              | Init complete, APPLID SNALKB03, TCPIP id TCPIPB
Line 2              | Maximum RU size is 00000600
Line 3     SNALKC04 | DLC path 00000001 pending
Line 4     SNALKC04 | Ready to accept bind from remote LU
Line 5     SNALKA04 | DLC path 00000002 pending
Line 6     SNALKA04 | Sending BIND request for SNA send session
Line 7     SNALKA04 | OPNDST   CHECK err. R15 00000004 R0 00000010 RTNCD 00000010 FDBK2 00000000
Line 8     SNALKA04 | OPNDST   sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 9     SNALKA04 | DLC path 00000002 pending
Line 10    SNALKA04 | Sending BIND request for SNA send session
Line 11    SNALKA04 | OPNDST   CHECK err. R15 00000004 R0 00000010 RTNCD 00000010 FDBK2 00000000
Line 12    SNALKA04 | OPNDST   sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 13    SNALKC04 | Received BIND request for SNA receive session
Line 14    SNALKC04 | Sending BIND request for SNA send session
Line 15    SNALKC04 | SNA receive session established
Line 16    SNALKC04 | SNA send session established
Line 17    SNALKC04 | Accepting DLC path 00000001
Line 18    SNALKA04 | DLC path 00000002 pending
Line 19    SNALKA04 | Sending BIND request for SNA send session
Line 20    SNALKA04 | SNA send session established
Line 21    SNALKA04 | Accepting DLC path 00000002
Line 22    SNALKA04 | Received BIND request for SNA receive session
Line 23    SNALKA04 | SNA receive session established
Line 24    SNALKC04 | NSEXIT CLEANUP request for receive session
Line 25    SNALKC04 | RECEIVE  CHECK err. R15 00000004 R0 0000000C RTNCD 0000000C FDBK2 0000000B
Line 26    SNALKC04 | RECEIVE  sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 27    SNALKC04 | DLC path 00000001 pending
Line 28    SNALKC04 | Ready to accept bind from remote LU
Line 29    STOP SNALINK
Line 30             | Received STOP command, shutting down
```

Figure 22. SNALINK Console Example

Line Number	Description
Lines 1 and 2	SNALINK displays its startup information from its command line parameters, which are customized as described in *z/OS Communications Server: IP*

	Configuration Reference. The maximum RU size and all other values are displayed in hexadecimal.
Lines 3 and 4	The TCPIP address space, TCPIPB, issues a DLC CONNECT to establish a session with the remote LU SNALKC04. SNALKC04 is higher in the collating sequence than the local LU name SNALKB03. Consequently, SNALKB03 takes the passive role in connecting to SNALKC04, and waits for SNALKC04 to establish a session.
Lines 5 and 6	TCP/IP issues another DLC CONNECT to establish a session with SNALKA04. In this case, SNALKA04 is lower in the collating sequence. Consequently, SNALKB03 takes an active role in connecting to SNALKA04.
Lines 7 and 8	The session establishment attempt to SNALKA04 has failed, as indicated by the (nonzero) return code and the sense information printed.
Lines 9 through 12	Thirty seconds later, TCP/IP again tries to connect to SNALKA04.
Lines 13 and 14	SNALINK receives a BIND request from SNALKC04. SNALINK calls the resulting session the receive session, because it is used only to send data from SNALKC04. Now that the active end has initiated communication, SNALKB03 as the passive end, sends a BIND request to establish a send session.
Lines 15 through 17	The send and receive sessions are fully established. Establishment of the send session causes SNALINK to accept the corresponding DLC path.
Lines 18 through 23	TCP/IP again tries to connect to SNALKA04. This time it is successful (success is indicated by no nonzero return codes).
Lines 24 through 26	SNALKC04 terminates its sessions, and various error messages result.
Lines 27 and 28	Thirty seconds later, TCP/IP again tries to establish communication with SNALKC04. As in lines 13 and 14, SNALKB03 is the passive partner.
Lines 29 and 30	The operator issues a STOP SNALINK command, which causes SNALINK to stop. ***All*** DLC paths and SNA sessions are ended.

Verifying Connection Status Using NETSTAT DEVLINKS

The DLC connect protocol between TCP/IP and SNALINK causes the status of the SNAIUCV device, reported by NETSTAT DEVLINKS, to reflect the status of the SNA sessions to the remote LU.

Status Reported	Explanation
`Issued connect`	Passive side: SNALINK is waiting for a remote LU to establish a session.

	Active side: SNALINK is trying to establish a session with a remote LU.
Will retry connect	The last session was ended, or the last session attempt failed. SNAIUCV driver retries the connection within 30 seconds.
Connected	An SNA send session is established. Under normal conditions this also means a receive session is established or will be established soon, and communication between the two LUs is possible.
Sending message	An SNA send session is established, and there is a DLC SEND currently outstanding.

Controlling the SNALINK LU0 Interface with the MODIFY Command

Both of the following commands would pass parameters to a SNALINK LU0 address space started with a procedure named SNLK12.TCPSETUP.

```
MODIFY SNLK12.TCPSETUP,HALT

F SNLK12.TCPSETUP,PKTTRACE CLEAR *
```

TCP/IP for MVS allows the configuration of multiple DLC links to the SNALINK LU0, LU6.2, and X.25 NPSI server address spaces. The PKTTRACE parameter supports this capability through the LINKNAME parameter. Multiple PKTTRACE parameters can be issued to define the scope of the tracing by identifying the tracing options applicable to multiple links.

PKTTRACE considerations:

- Parsing of the parameter halts as soon as an error is detected and the parameter is ignored.
- Parameters can appear in any order.
- The occurrence of a parameter more than once is an error. In the case of the special parameters ON, OFF, CLEAR, and LIST, the occurrence of more than one of these parameters is an error.
- The PKTTRACE parameter must be issued after the corresponding DLC connection has been accepted from TCPIP.
- Each defined link will have an associated trace profile. The trace profile stores the effective values of each of the trace options for the link. When created or reset using the CLEAR parameter, a link's trace profile is set to the default values for the trace parameters as follows:
  DESTPORT
  No checking
  FULL Tracing of the whole IP packet
  IP All IP addresses (*)
  PROT All protocols (*)
  SRCPORT
  No checking
  SUBNET
  No checking
- Multiple statements can refer to the same link either by explicitly naming the link or by defaulting to an asterisk (*), which indicates all links. When multiple statements refer to the same link, the parameters on the statements are cumulative, and parameters not specified on the second and subsequent statements are not changed. If a parameter is specified on one statement and

then appears on a subsequent statement, the value associated with the last occurrence of the option is used because this is the value that is stored in the trace.

SNALINK LU6.2

The SNALINK LU6.2 cataloged procedure runs a VTAM application program called SNALNK62, which is an interface between the TCPIP address space and the SNA network. SNALNK62 uses SNA LU type 6.2 sessions to pass the TCP/IP data to or from SNALNK62 devices running on other hosts. Examples of SNALNK62 devices include an OS/2 workstation running TCP/IP for OS/2 or a host running TCP/IP for MVS.

Configuring SNALINK LU6.2

Steps to configure SNALINK LU6.2:

1. Specify DEVICE and LINK statements in *hlq*.PROFILE.TCPIP.
2. Update the SNALINK LU6.2 cataloged procedure.
3. Define the SNALINK LU6.2 application to VTAM.
4. Update the SNALINK LU6.2 configuration data set.

Step 1: Specify DEVICE and LINK Statements in hlq.PROFILE.TCPIP

You must update the *hlq*.PROFILE.TCPIP data set to include a DEVICE and LINK statement for each DLC connection to be established between the main TCPIP address space and the SNALINK LU6.2 address space.

Step 2: Update the SNALINK LU6.2 Cataloged Procedure

Update the SNALINK LU6.2 cataloged procedure by copying the sample in SEZAINST(LU62PROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. No system parameters are required for the SNALINK LU6.2 address space.

The DD statements in the cataloged procedure should be defined as follows:

DD Name	Description
SYSTCPD	TCPIP.DATA configuration data set
LU62CFG	SNALINK LU6.2 configuration data set
SYSPRINT	Runtime diagnostic or trace output
SYSUDUMP	User abend dump output (optional)

Note: SNALINK LU6.2 does not use the default search sequence to find TCPIP.DATA. Therefore, the //SYSTCPD statement is required.

Step 3: Define the SNALINK LU6.2 Application to VTAM

SNALINK LU6.2 opens two SNA LU type 6.2 sessions with each destination node; one for sending and one for receiving. If a destination node supports parallel SNA LU type 6.2 sessions (PARSESS=YES), the two sessions use the same remote logical unit; otherwise, two remote logical units are used. In either case, SNALINK LU6.2 uses a single local logical unit that must support parallel sessions.

The SNALINK LU6.2 address space must be defined to VTAM as an SNA LU type 6.2 application program. The following APPL statement defines a SNALINK LU6.2 application to VTAM.

```
LU62APPL  APPL ACBNAME=LU62APPL,                                      *
               PRTCT=QWERTY,                                          *
               AUTH=(ACQ,VPACE),                                      *
               SRBEXIT=NO,                                            *
               EAS=12,                                                *
               PARSESS=YES,                                           *
               SONSCIP=YES,                                           *
               APPC=YES,                                              *
               DLOGMOD=LU62MODE,                                      *
               VPACING=0
```

Figure 23. APPL Statement for SNALINK LU6.2

Note: ***SRBEXIT must be NO.***

See *z/OS Communications Server: SNA Resource Definition Reference* for further information about defining VTAM applications.

The LOGMODE table entry specified by the APPL DLOGMOD parameter should have the following form:

```
LU62MODE MODEENT LOGMODE=LU62MODE,FMPROF=X'13',TSPROF=X'07',          *
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',           *
               RUSIZES=X'8585',ENCR=B'0000',                          *
               PSERVIC=X'060200000000000000000300'
```

See *z/OS Communications Server: SNA Customization* for more information about defining log mode tables and *OS/390 IBM Communications Server: SNA Programming* for information on PSERVIC values.

Step 4: Update the SNALINK LU6.2 Configuration Data Set

Customize the SNALINK LU6.2 configuration data set by copying the sample provided in SEZA.INST(LU62CFG) to your system or recognized PROCLIB and modifying it to suit your local conditions. Add or change the configuration statements as required. Be sure the //LU62CFG statement in the cataloged procedure points to this data set. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about parameters.

X.25 NCP Packet Switching Interface (NPSI)

The X.25 NPSI server runs a VTAM application program called XNX25IPI, which is the interface between the TCPIP address space's DLC driver and your X.25 network. XNX25IPI communicates with the X.25 NCP Packet Switching Interface in a front-end IBM 37xx Communications Controller.

Large scale X.25 network applications often require multiple physical lines to the network switch for increased capacity and reliability. You can configure the X.25 NPSI server to support multiple lines as a group, rather than individually. In this configuration, the collection of lines is assigned a single address called a ***hunt group*** address. Incoming X.25 calls are distributed among the lines in either rotary or traffic balancing fashion, depending on the services offered by the X.25 network provider.

For information about improving the performance of the X.25 NPSI network, see the options on the PORT statement and GATEWAY statement in the *hlq*.PROFILE.TCPIP and the explanation provided in the *z/OS Communications Server: IP Configuration Reference*.

Configuring X.25 NPSI

This section describes how to configure the X.25 NPSI server.

Steps to Configure the X.25 NPSI Server:

1. Specify X.25 configuration statements in *hlq*.PROFILE.TCPIP.
2. Update the X.25 NPSI cataloged procedure.
3. Update the X.25 NPSI server configuration data set.
4. Define the X.25 NPSI configuration.
5. Define the X.25 NPSI application to VTAM.
6. Define VTAM Switched Circuits.

If you want to run the X.25 NPSI cataloged procedure in a different domain than the X.25 NPSI communication controller, see *z/OS Communications Server: IP Configuration Reference*.

For information about operating the X.25 NPSI server with the MODIFY command, see *z/OS Communications Server: IP Configuration Reference*.

Step 1: Specify X.25 Configuration Statements in hlq.PROFILE.TCPIP

To configure the *hlq*.PROFILE.TCPIP data set for X.25 NPSI, include appropriate DEVICE, LINK, HOME, GATEWAY, and START statements. The following example shows the statements that would correspond with the other X.25 samples in this chapter.

```
;
DEVICE X25DEV X25NPSI TCPIPX25
LINK X25LINK SAMEHOST 1 X25DEV
;
HOME
   199.005.058.23     X25LINK
;
GATEWAY
;
; Network  First hop   Link name Packet size  Subnet mask  Subnet value
  192.005      =        X25LINK     2000       0.0.255.0    0.0.58.0
;
START X25DEV
;
```

Note: Only one DEVICE and LINK statement per TCPIPX25 address space is allowed.

Step 2: Update the X.25 NPSI Cataloged Procedure

Update the X.25 NPSI cataloged procedure by copying the sample provided in SEZAINST(X25PROC) to your system or recognized PROCLIB and modifying it to suit your local conditions.

Change the data set names as needed:

- The X.25 interface does not use the TCP/IP default search sequence for *hlq*.TCPIP.DATA, therefore //SYSTCPD DD is required.
- Modify the //X25IPI DD statement to point to your X.25 configuration data set.

Step 3: Update the X.25 NPSI Server Configuration Data Set

A sample configuration data set provided in SEZAINST(X25CONF) gives examples of how to define a public network connection, a Defense Data Network connection, and private point-to-point connection to a router. Copy this sample to the data set

pointed to by the //X25IPI DD statement in your X.25 NPSI cataloged procedure. Update this sample to define your X.25 connections using the statements listed in the *z/OS Communications Server: IP Configuration Reference*.

Each connection must have a LINK and at least one DEST statement. You can optionally define hunt groups, fast connects, and call handling options for each link, and global options such as trace levels, when to clear inactive connections, and the buffer size to use for IP datagrams. You can find complete syntax for each of these statements in *z/OS Communications Server: IP Configuration Reference*.

Step 4: Define the X.25 NPSI Configuration

Define the X.25 NPSI configuration according to the information in *X.25 NPSI Planning and Installation*. The X.25 NPSI server supports use of the LOGAPPL operand on the X25.MCH definition in the X.25 NPSI configuration to allow automatic recovery. You can use either the Generalized Access to X.25 Transport Extension (GATE) or Dedicated Access to X.25 Transport Extension (DATE).

IBM recommends using the X.25 NPSI GATE configuration which allows sharing of an X.25 physical link and provides better error recovery. A sample is provided in SEZAINST(NPSIGATE). NPSI GATE requires that you include the OPTIONS GATE statement in the X.25 NPSI configuration data set after the LINK statement, as shown in this portion of the X25CONF sample:

```
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU024     PRIV 1               2      1024   2
Options  GATE
*
*        IP address      X.25 DTE addr       C.U.D.
*        --------------- ---------------     --------
Dest     192.5.57.2      2
```

Sites that need to use the X.25 NPSI DATE configuration can find a sample in PV SEZAINST(NPSIDATE). See *X.25 NPSI Host Programming* for information about the definitions and parameters used in these configurations.

The following example shows portions of the sample NPSI GATE configuration (NPSIGATE). Ellipses (....) indicate code that has been omitted.

```
************************************************************************
         OPTIONS NEWDEFN=YES,USERGEN=X25NPSI
************************************************************************
.....
.....
 NPSIV32  BUILD ADDSESS=400,                                           +
                AUXADDR=800,                                           +
                ERLIMIT=16,                                            +
                NAMTAB=120,                                            +
                MAXSESS=250,                                           +
                USGTIER=5,                                             +
                BRANCH=8000,                                           +
                BFRS=104,           BUFFER SIZE TO BE GENED            +
                CATRACE=(YES,255),  CHAN.ADAPTER TRACE OPTION          +
                CSMSG=C3D9C9E340E2C9E340D4C5E2E2C1C7C540C6D6D940E2E24040+
                40C2C340E3C5D9D4C9D5C1D3,                              +
                CWALL=26,                                              +
                ENABLTO=30.0,                                          +
                ERASE=YES,                                             +
                LOADLIB=NCPLOAD,    TARGET OF FINAL LINKEDIT           +
                LTRACE=8,           LINES TRACED SIMULTANEOUSLY        +
```

```
               MAXSSCP=8,          NUMBER OF CONCURRENT SSCP'S         +
               MODEL=3745,                                             +
               VERSION=V5R2.1,                                         +
               NEWNAME=NPSITCP,    NAME OF NCP LOAD MODULE             +
               NUMHSAS=8,          HOST SA IN CONCURRENT COMMUNICATION +
               OLT=YES,            ONLINE TERMINAL TEST                +
               PWROFF=YES,                                             +
               BACKUP=500,                                             +
               SALIMIT=511,                                            +
               SLODOWN=12,         BUFFER SLOWDOWN THRESHOLD (PERCENT) +
               SUBAREA=03,                                             +
               TRACE=(YES,100),    ADDRESS TRACE OPTION IN CORE TABLE  +
               TYPGEN=NCP,                                             +
               TYPSYS=MVS,         NCP TO BE GENERATED ON MVS          +
               TWXID=(E8D6E4C3C1D3D311,C2C9C7D5C3D7C3C1D3D325),        +
               VRPOOL=30,                                              +
               TRANSFR=32,                                             +
               NETID=NETA,                                             +
               X25.USGTIER=5,                                          +
               X25.IDNUMH=01,                                          +
               X25.MCHCNT=4,                                           +
               X25.MAXPIU=64K
.....
.....
*************************************************************************
*
*                  NPSI DEFINITIONS
*
*************************************************************************
X25XXX   X25.NET CPHINDX=1,                                            +
               NETTYPE=1,                                              +
               DM=YES,                                                 +
               OUHINDX=1
         X25.VCCPT INDEX=1,                                            +
               MAXPKTL=128,                                            +
               VWINDOW=2
         X25.OUFT INDEX=1
*-----------------------------------------------------------------------*
*        Hunt group primary line 021 with fast connect                 *
*-----------------------------------------------------------------------*
HGRP01A  X25.MCH ADDRESS=21,                                           +
               FRMLGTH=131,        128 byte packet + 3 byte header     +
               PKTMODL=8,                                              +
               ANS=CONT,                                               +
               LCGDEF=(0,16),      16 logical channels in group 0      +
               MWINDOW=2,                                              +
               STATION=DTE,                                            +
               SPEED=9600,                                             +
               LCN0=NOTUSED,                                           +
               GATE=GENERAL,       GATE                                +
               LLCLIST=(LLC4),                                         +
               CONNECT=YES,        Fast connect                        +
               LOGAPPL=TCPIPX25,                                       +
               DBIT=NO,                                                +
               DIRECT=NO,                                              +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,16),                                            +
               MAXDATA=1034,       MAXDATA only with Fast connect!     +
               TYPE=SWITCHED,                                          +
               CALL=INOUT,                                             +
               OUFINDX=1,                                              +
               VCCINDX=1
*-----------------------------------------------------------------------*
*        Hunt group secondary line 022 with fast connect               *
*-----------------------------------------------------------------------*
HGRP01B  X25.MCH ADDRESS=22,                                           +
```

```
                FRMLGTH=131,        128 byte packet + 3 byte header     +
                PKTMODL=8,                                              +
                ANS=CONT,                                               +
                LCGDEF=(0,16),      16 logical channels in group 0      +
                MWINDOW=2,                                              +
                STATION=DTE,                                            +
                SPEED=9600,                                             +
                LCN0=NOTUSED,                                           +
                GATE=GENERAL,       GATE                                +
                LLCLIST=(LLC4),                                         +
                CONNECT=YES,        Fast connect                        +
                LOGAPPL=TCPIPX25,                                       +
                DBIT=NO,                                                +
                DIRECT=NO,                                              +
                SUBADDR=NO
          X25.LCG LCGN=0
          X25.VC LCN=(1,16),                                            +
                MAXDATA=1034,       MAXDATA only with Fast connect!     +
                TYPE=SWITCHED,                                          +
                CALL=INOUT,                                             +
                OUFINDX=1,                                              +
                VCCINDX=1
*-------------------------------------------------------------------------*
*       DDN line 023                                                      *
*-------------------------------------------------------------------------*
DTE01   X25.MCH ADDRESS=23,                                             +
                FRMLGTH=131,        128 byte packet + 3 byte header     +
                PKTMODL=8,                                              +
                ANS=CONT,                                               +
                LCGDEF=(0,16),      16 logical channels in group 0      +
                MWINDOW=2,                                              +
                STATION=DTE,                                            +
                SPEED=9600,                                             +
                LCN0=NOTUSED,                                           +
                GATE=GENERAL,       GATE                                +
                LLCLIST=(LLC4),                                         +
                LOGAPPL=TCPIPX25,                                       +
                CTCP=(00),          paired with CUD list                +
                CUD0=(CC),          incoming CUD selects CTCP           +
                DBIT=NO,                                                +
                DIRECT=NO,                                              +
                SUBADDR=NO
          X25.LCG LCGN=0
          X25.VC LCN=(1,16),                                            +
                TYPE=SWITCHED,                                          +
                CALL=INOUT,                                             +
                OUFINDX=1,                                              +
                VCCINDX=1
*-------------------------------------------------------------------------*
*       Private line 024: DCE station to router                           *
*-------------------------------------------------------------------------*
DCE01   X25.MCH ADDRESS=24,         1024 byte packet + 3 byte header    +
                FRMLGTH=1027,                                           +
                PKTMODL=8,                                              +
                ANS=CONT,                                               +
                LCGDEF=(0,2),                                           +
                MWINDOW=2,                                              +
                STATION=DCE,                                            +
                SPEED=9600,                                             +
                LCN0=NOTUSED,                                           +
                GATE=GENERAL,                                           +
                LLCLIST=(LLC4),                                         +
                CTCP=(00),          paired with CUD list                +
                CUD0=(CC),          incoming CUD selects CTCP           +
                DBIT=NO,                                                +
                DIRECT=NO,                                              +
                SUBADDR=NO
```

```
         X25.LCG LCGN=0
         X25.VC LCN=(1,2),                                              +
               TYPE=SWITCHED,                                           +
               CALL=INOUT,                                              +
               OUFINDX=1,                                               +
               VCCINDX=1
         X25.END
 ************************************************************************
....
....
 GENEND   GENEND
```

Step 5: Define the X.25 NPSI Application to VTAM

Define the X.25 NPSI VTAM application with an APPL statement in VTAMLST. Following is an example of a VTAM APPL statement for X.25 NPSI.

```
         VBUILD TYPE=APPL                                               +
 TCPIPX25 APPL  ACBNAME=TCPIPX25,                                       +
               PRTCT=TCPX25,                                            +
               AUTH=(ACQ),                                              +
               PARSESS=YES,                                             +
               EAS=20
```

Step 6: Define VTAM Switched Circuits

X.25 NPSI switched virtual circuits (SVCs) appear to VTAM as switched links, [1] requiring a switched circuit definition of a physical unit (PU) and logical unit (LU) for each SVC. The sample provided in SEZAINST(X25VSVC) shows the definitions of a VTAM switched circuit corresponding to the sample X.25 NPSI GATE configuration.

The definitions are associated with the SVCs by identifying numbers (IDNUMs) created automatically during X.25 NPSI generation. The entries, in hexadecimal, run in steps of 2, by default, in the opposite order of the MCH and SVC definitions in the X.25 NPSI configuration.

Notes:

1. Permanent virtual circuits (PVCs) are not supported.
2. If you specify a local version of the z/OS UNIX table with the SSCPFM operand, the table must not have an entry for message 10 (the welcome message); otherwise, the X.25 NPSI server does not operate correctly.

Following is a sample SVC configuration data set (X25VSVC):

```
         VBUILD TYPE=SWNET,MAXGRP=1,MAXNO=1
 *----------------------------------------------------------------------*
 *   Switched circuits for DDN line 023 (16 VCs, IDNUMS 006-024)       *
 *                                                                      *
 *   COPYRIGHT = NONE.                                                  *
 *----------------------------------------------------------------------*
 VP023001 PU    ADDR=23,IDBLK=003,IDNUM=01024,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
 VL023001 LU    LOCADDR=0
 VP023002 PU    ADDR=23,IDBLK=003,IDNUM=01022,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
 VL023002 LU    LOCADDR=0
 VP023003 PU    ADDR=23,IDBLK=003,IDNUM=01020,                          +
                DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
                SSCPFM=USSNTO
```

1. Except when using fast connect, where they appear as leased lines to VTAM. For more information, see *z/OS Communications Server: IP Configuration Reference*.

```
VL023003 LU    LOCADDR=0
VP023004 PU    ADDR=23,IDBLK=003,IDNUM=0101E,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023004 LU    LOCADDR=0
VP023005 PU    ADDR=23,IDBLK=003,IDNUM=0101C,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023005 LU    LOCADDR=0
VP023006 PU    ADDR=23,IDBLK=003,IDNUM=0101A,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023006 LU    LOCADDR=0
VP023007 PU    ADDR=23,IDBLK=003,IDNUM=01018,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023007 LU    LOCADDR=0
VP023008 PU    ADDR=23,IDBLK=003,IDNUM=01016,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023008 LU    LOCADDR=0
VP023009 PU    ADDR=23,IDBLK=003,IDNUM=01014,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023009 LU    LOCADDR=0
VP023010 PU    ADDR=23,IDBLK=003,IDNUM=01012,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023010 LU    LOCADDR=0
VP023011 PU    ADDR=23,IDBLK=003,IDNUM=01010,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023011 LU    LOCADDR=0
VP023012 PU    ADDR=23,IDBLK=003,IDNUM=0100E,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023012 LU    LOCADDR=0
VP023013 PU    ADDR=23,IDBLK=003,IDNUM=0100C,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023013 LU    LOCADDR=0
VP023014 PU    ADDR=23,IDBLK=003,IDNUM=0100A,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023014 LU    LOCADDR=0
VP023015 PU    ADDR=23,IDBLK=003,IDNUM=01008,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023015 LU    LOCADDR=0
VP023016 PU    ADDR=23,IDBLK=003,IDNUM=01006,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023016 LU    LOCADDR=0
*-------------------------------------------------------------------*
*   Switched circuits for private line 024 (2 VCs, IDNUMS 002-004)  *
*-------------------------------------------------------------------*
VP024001 PU    ADDR=24,IDBLK=003,IDNUM=01004,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL024001 LU    LOCADDR=0
VP024002 PU    ADDR=24,IDBLK=003,IDNUM=01002,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL024002 LU    LOCADDR=0
```

NCPROUTE

NCPROUTE is a server that provides an alternative to using the Network Control Program (NCP) as a static host-independent IP router. NCPROUTE has the following effects:

- NCP becomes an active RIP router on a TCP/IP network
- NCP becomes responsive to SNMP route table queries

Notes:

1. NCPROUTE requires NCP V7R1, or later.
2. NCPROUTE requires SNALINK LU0 when using NCP V7R3 or previous.
3. SNALINK and IP over CDLC is supported for ESCON®, BCCA, and CADS channels.
4. IP over CDLC can be used instead of SNALINK when using NCP V7R4, or later.
5. If using RIP Version 2, NCPROUTE requires NCP V7R6, or later. Also, the NCP generation definition must have VSUBNETS=YES specified on the BUILD statement.
6. NCP versions V6R1 and V6R2 support static IP routing only. NCP uses these static route tables to deliver datagrams over connected TCP/IP networks. NCP V7R1 can be specified only as a host-dependent router and it requires the NCPROUTE server to function as a RIP router.
7. If using NCPROUTE with SNALINK, IP over CDLC channels, and OROUTED, you should customize the NCST interface metric on the NCP client side for the SNALINK NCST connection so the routes will be less preferred. This causes RouteD to prefer routes from the IP over CDLC interface over the ones from the SNALINK interface. To customize the interface metric, see the *interface metric* option in “Step 8: Configure the NCPROUTE Gateways Data Set (Optional)” on page 219. Do the same for the SNALINK interface on the MVS host side by customizing the metric in the BSDROUTINGPARMS statement. RIP traffic will be carried over the IP over CDLC interface, while transport PDUs (for example, Hello, Add Route Request, Delete Route Request) will be carried over the SNALINK interface.
8. NCPROUTE does not support zero subnets.

NCPROUTE provides dynamic route table updates for one or more NCP clients that have been generated as IP routers and have NCPROUTE specified as the NCPROUTE server. NCPROUTE tables are updated periodically in the NCP client based on updates sent by the NCPROUTE server. These updates reflect dynamic changes in route states.

An NCPROUTE server at the host uses the Routing Information Protocol (RIP), described in RFC 1058 (RIP version 1) and in RFC 1723 (RIP version 2). The same routing protocols are used by the OROUTED server. NCPROUTE is implemented as a RIP server operating on an MVS host connected to a RIP client in the NCP. Together they provide the appearance to the TCP/IP network of an IP router using the RIP protocol. The same client/server pair also provides SNMP agent support for network management route table queries. RIP Versions 1 and 2 are currently supported by NCPROUTE. For a brief description of RIP (Versions 1 and 2), see “Chapter 4. Routing” on page 143.

Understanding the NCPROUTE Environment

The NCPROUTE server:

- Supports multiple host-attached, link-attached, and remote link-attached NCP clients as illustrated in Figure 24
- Generates RIP datagrams for the NCP to send
- Maintains separate routing tables for each NCP client
- Generates SNMP route table responses for each NCP SNMP agent

The client NCP unit appears as an active router to other RIP routers on the network. Multiple NCP clients can connect to the same NCPROUTE server. Each NCP appears as an IP router to the rest of the network. Each NCP client must have one or more LU0 sessions established with SNALINK. One LU0 session per client is used as the primary session, with the remaining sessions serving as backups.

Figure 24 illustrates the different ways the NCPROUTE server can support NCP clients. NCP3 and NCP4 are host-attached NCP clients, NCP5 and NCP6 are link-attached NCP clients, and NCP1 and NCP2 are remote link-attached NCP clients.

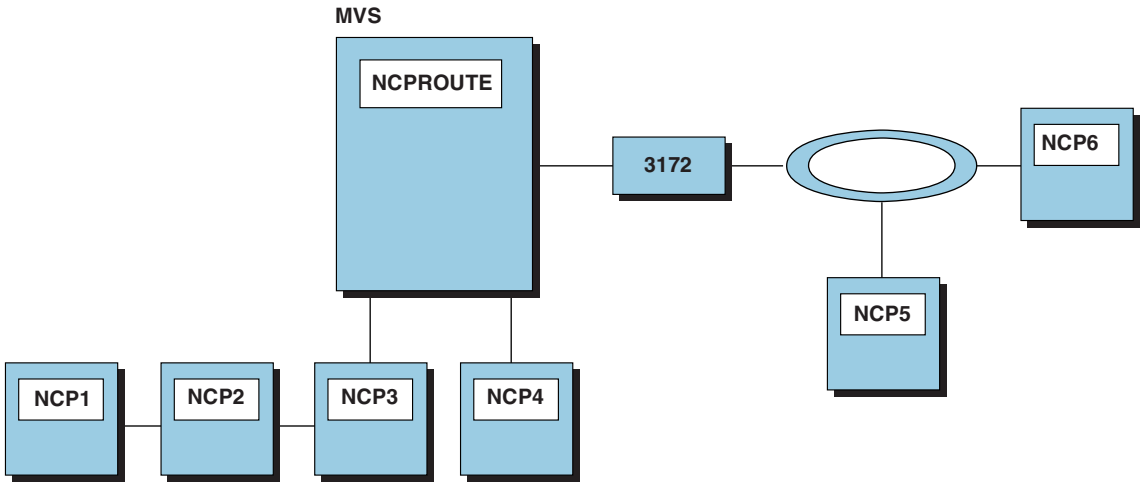


Figure 24. NCPROUTE Environment

Server Requirements

NCPROUTE processes RIP and SNMP datagrams addressed to all attached NCP units, generates datagrams for the NCP units, and maintains the state of each NCP unit's routing tables.

SNMP support is limited to route table queries. Queries are made to the NCP, which sends the request to the NCPROUTE server for processing.

NCPROUTE Operation

An NCP's IPOWNER statement defines the controlling host and the interface this NCP client must use to reach the host. The NCP client initiates contact with NCPROUTE by sending a datagram, known as a "Hello" message, to the controlling host. It transmits this datagram on UDP port 580.

Note: The port number is generated in the NCP (using the UDPPORT keyword on the IPOWNER statement) and configured in NCPROUTE.

The “Hello” message identifies the client NCP and determines which member from the *hlq*.NCPROUTE.GATEWAYS partitioned data set to use for this NCP’s route table. Any valid MVS data set name can be used for the gateways data set.

The NCP client then sends a list of its inactive links to NCPROUTE. NCPROUTE uses additional routes defined for this NCP in the NCPROUTE gateways data set, as defined in the NCPROUTE profile. It also uses the inactive links provided dynamically by the NCP to build the current route table for this NCP. The following process is repeated for each NCP that has been generated to act as a RIP router:

1. A RIP packet arrives at the NCP client from a foreign router.
2. The NCP client sends this datagram to the NCPROUTE server.
3. The NCPROUTE server processes the RIP packet.
4. The NCPROUTE server creates a RIP update for an NCP client.
5. This update is sent to the NCP client.
6. The NCP client transmits the datagram to the network.

NCPROUTE sends route table updates to each NCP client every 30 seconds. After a client has been activated, updates must be supplied over each of its interfaces every 30 seconds. The NCPROUTE server creates these updates and sends them to the NCP client along with the IP addresses of other RIP routers that the NCP client should send them to.

At the same time, adjacent RIP routers are providing periodic updates every 30 seconds to NCPROUTE. These updates are sent by the NCP client to the NCPROUTE server, where they are processed, and the results are reflected in future updates back to the NCP client.

The NCP client sends all SNMP and RIP datagrams to the NCPROUTE server for processing. The NCPROUTE server provides RIP packets and SNMP replies to the NCP client to send to their final destination.

NCPROUTE Gateways:

Passive RIP Route: Information about passive routes is put in NCP’s and NCPROUTE’s routing tables. A passive entry in NCPROUTE’s routing table is used as a placeholder to prevent a route from being propagated and from being overwritten by a competing RIP route. With the exception of directly-connected passive routes, passive routes are not propagated; they are known only by this router.

Using passive routes can create routing loops, so care must be taken when creating them.

Do not define passive routes such as these:

- A to C is via B
- B to C is via A

Passive routes should be used when adding routes where the host or network is not running RIP. Passive routes should also be used when adding a default route, because this is the only way to prevent a route from timing out.

External RIP Route: External routes are managed by other protocols, for example, the External Gateway Protocol (EGP). NCPROUTE needs to know not to interfere with these routes and not to delete them.

An external entry exists in the NCPROUTE routing table as a place holder to prevent a route from being overwritten by a competing RIP route. External routes are not propagated. NCPROUTE does not manage an external route. Therefore, NCPROUTE only knows that there is an existing route to the host or network and that is the route known by NCP.

External routes should be used when the local machine is running a non-RIP routing protocol that dynamically changes the TCP/IP routing tables. The remote machine does not need to run any routing protocol, since the only concerns are how to route traffic from the local machine to the remote machine, and how to prevent multiple routing protocols from interfering with each other.

RIP Route Advertising Rules: Table 13 illustrates the differences between routing rules on the basis of RIP version.

Table 13. RIP Route Advertising Rules

Version[2]	**Advertised Destination route[1]**	**Same Subnet as Interface**	**Different Network from Interface with Same Subnet Mask**	**Same Network as Interface Regardless of Subnet Mask**	**Different Network from Interface**	**Same Supernet as Interface**	**Different Supernet from Interface**
RIPv1	Host	Yes[3]	Yes[3]	Yes[3]	Yes[3]		
	Subnet	No	Yes	No	No		
	Network			No	Yes		
	Supernet						
	Default				Yes[3]		
RIPv2	Host	Yes[3]	Yes[3]	Yes[3]	Yes[3]	Yes[3]	Yes[3]
	Subnet	No	Yes	Yes	Yes	Yes	Yes
	Network			No	Yes	No	Yes
	Supernet			No	Yes	No	Yes
	Default				Yes[5]		

Notes:

1. According to RIP design, route advertising relies on network-specific routes because they are the lowest common denominator. The network-specific routes consist of supernet, network, and subnet routes. The advertising of host specific routes is optional.
2. RIPv1 is the default setting for the RIP version. To set to RIPv2, specify the RIP2 parameter in NCPROUTE Profile and/or on interface options in the NCPROUTE Gateways data set.
3. The optional host specific routes are allowed to be advertised outside networks, and they are advertised in addition to the network specific routes. The option is enabled when the system -h parameter (or SUPPLY HOSTS option in NCPROUTE Gateways data set) is specified.

4. Although it is possible to advertise only the host specific routes using the RIP filters, doing so creates network unreachable problems when some routers in the network do not support the host specific routes. These routers rely on network-specific routes.
5. A default route has a network number of zero and is usually advertised over all network interfaces.
6. It does not matter whether the advertised route is VIPA or not. VIPA routes follow the same advertising rules as the non-VIPA routes.
7. Routes that are subjected to RIP filters may not be advertised at all over certain network interfaces.

NCPROUTE Active Gateways: Active gateways are treated as remote network interfaces. Active gateways are routers that are running RIP, but are reached through a medium that does not allow broadcasting or multicasting and is not point-to-point, for example, Hyperchannel. NCPROUTE normally requires that routers be reachable by broadcast or multicast for non-point-to-point links or by unicast addresses for point-to-point links. If the interface is neither, then an active gateway entry can add the gateway to NCPROUTE's interface list. NCPROUTE will treat the active gateway as a remote network interface. Note that the active gateway must be directly connected.

Active gateways should be used when the foreign router is reachable over a non broadcast-capable, non multicast-capable, and non point-to-point network, and is directly connected to the local host.

NCPROUTE communicates with active routes by unicast transmissions to the gateway address. Routes are not added immediately to either NCPROUTE or the NCP routing table. They are added and propagated normally when route advertisements arrive from an active gateway. The sole effect of an active gateway statement is to bypass the requirement for broadcast communication on non-point-to-point links. Interfaces that are not broadcast, non point-to-point, and are not active gateways are assumed to be loopback interfaces to the local machine. Also, while a route to an active gateway might timeout, the interface entry is never removed. If transmissions resume, then the new routes will still be available to the active gateways.

NCPROUTE Gateways Summary

Table 14 provides a list of NCPROUTE gateways and their characteristics.

Table 14. NCPROUTE Gateways Summary

	Propagate	Kernel	NCPROUTE	Timeout
Dynamic (1)	Yes	Yes	Yes	Yes
Passive	No (2)	Yes	Yes	No
External	No	No	Yes	No
Active	Yes	Yes	Yes	Yes

1 Dynamic routing is provided by NCPROUTE.

2 Except directly-connected passive routes. Directly-connected passive routes are propagated to other network interfaces for network reachability. A directly-connected passive route is one where the gateway address is one of the local interfaces in a NCP client.

RIP Input/Output Filters

The RIP input/output filters provide routing table manipulation and routing control. The filters are provided by NCPROUTE and consist of:

- Route receiving (unconditional and conditional)
- Route noreceiving
- Route forwarding (unconditional and conditional)
- Route noforwarding
- Interface Supply switch
- Interface RIP On/Off switch
- Gateway noreceiving

For more information on these filters, see “Step 8: Configure the NCPROUTE Gateways Data Set (Optional)” on page 219.

Configuring NCPROUTE

Steps to Configure NCPROUTE:

1. Specify configuration statements in *hlq*.PROFILE.TCPIP.
2. If using SNALINK, configure VTAM and SNALINK applications.
3. If using IP over CDLC, configure IP over CDLC DEVICE and LINK statements.
4. Update the NCPROUTE cataloged procedure.
5. Update *hlq*.ETC.SERVICES.
6. Configure the host-dependent NCP clients.
7. Configure the NCPROUTE profile data set for SNMP support and specification of an NCPROUTE gateways data set (optional).
8. Configure the NCPROUTE gateways data set for each NCP client (optional).
9. If RouteD is not used, define a directly-connected host route to each NCP client.

Figure 25 on page 211 shows the network addresses used in the configuration examples:

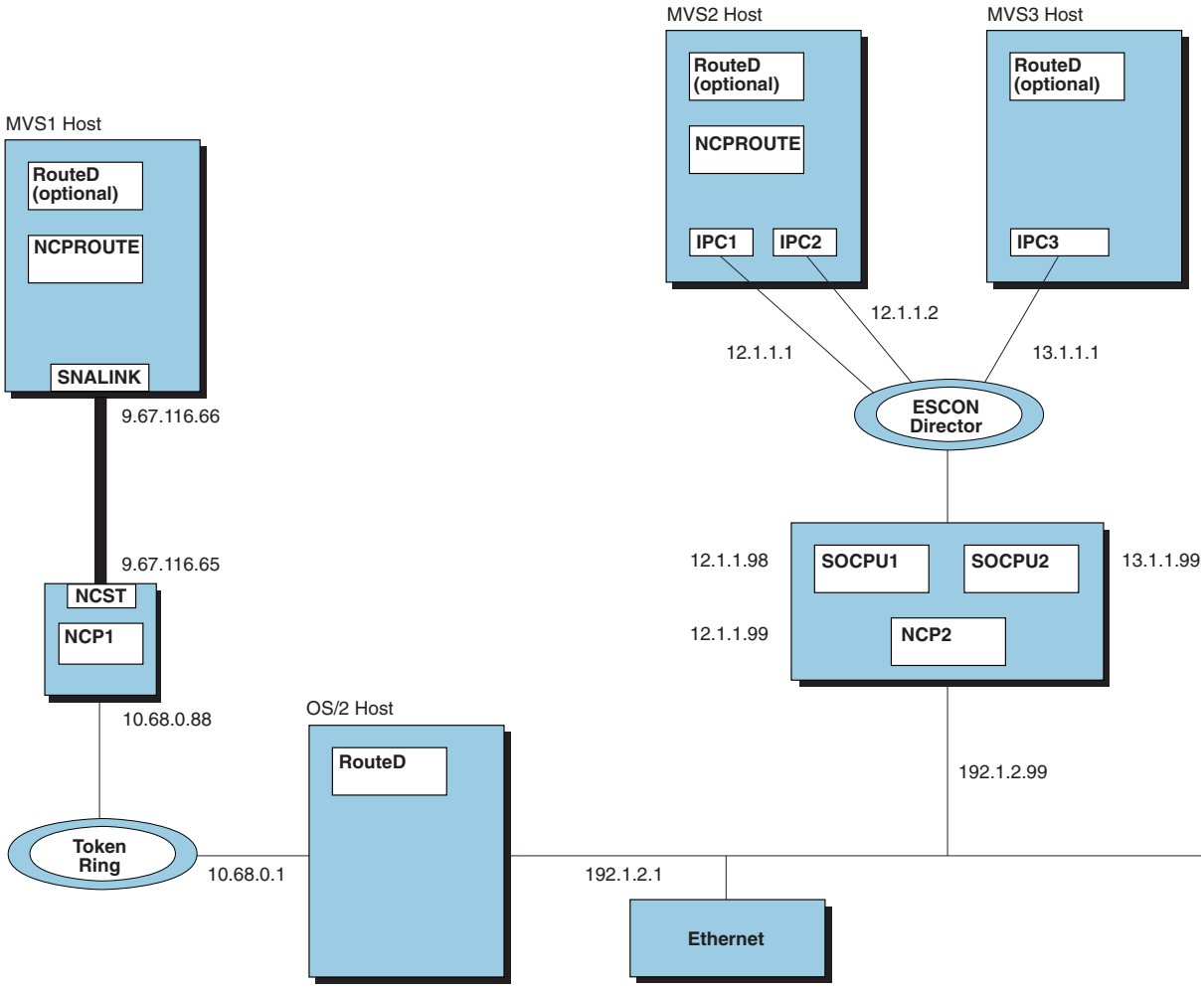


Figure 25. NCPROUTE Example Configuration

Step 1: Specify Configuration Statements in hlq.PROFILE.TCPIP

1. To have the NCPROUTE server started automatically when the TCPIP address space is started, include the name of the member containing the NCPROUTE cataloged procedure in the AUTOLOG statement in *hlq*.PROFILE.TCPIP:

   ```
   AUTOLOG
      NCPROUT
   ENDAUTOLOG
   ```

2. To ensure that UDP port 580 is reserved for the NCPROUTE server, also add the name of the member containing the NCPROUTE cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

   ```
   PORT
      580 UDP NCPROUT
   ```

   Note: This port number must match the one defined in the NCP generation definition (using the UDPPORT keyword on the IPOWNER statement) and assigned in *hlq*.ETC.SERVICES.

3. NCPROUTE also requires HOME and BSDROUTINGPARMS statements for the SNALINK type LU0 and IP over CDLC connections. For example, you would use this HOME and BSDROUTINGPARMS statement and, optionally, the GATEWAY statement for the configuration shown in Figure 25:

```
MVS1:   HOME
           9.67.116.66  SNALINK
           BSDROUTINGPARMS false
             SNALINK 2000     0    255.255.240.0    9.67.116.65
           ENDBSDROUTINGPARMS
MVS2:   HOME
           12.1.1.1   IPC1
           12.1.1.2   IPC2
           BSDROUTINGPARMS false
             IPC1    1000     0    255.255.255.0   12.1.1.98
             IPC2    1000     0    255.255.255.0   12.1.1.99
           ENDBSDROUTINGPARMS
MVS3:   HOME
           13.1.1.1   IPC3
           BSDROUTINGPARMS false
             IPC3    1000     0    255.255.255.128   13.1.1.99
           ENDBSDROUTINGPARMS
```

Notes:

a. If you are not using RouteD to manage the host routes, configure static routes to the NCP client or clients in the GATEWAY statement in *hlq*.PROFILE.TCPIP. If using NCPROUTE with OMPROUTE, BSDROUTINGPARMS is required to route Transport PDUs prior to OMPROUTE activation. Since the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in both BSDROUTINGPARMS statement and the OMPROUTE configuration file. See “Step 9: Define a Directly Connected Host Route for the NCST Session” on page 223 for sample definition. For more information on the GATEWAY statement, see *z/OS Communications Server: IP Configuration Reference* for each NCP client.

b. A BSDROUTINGPARMS statement is required even though RouteD is not used.

You can find a complete explanation of these configuration statements in *z/OS Communications Server: IP Configuration Reference*.

Step 2: Configure VTAM and SNALINK Applications

If you are using NCP V7R3 or previous, NCPROUTE requires SNALINK type LU0 to run. To use SNALINK LU0, verify that you have configured the SNALINK LU0 interface, defined it to VTAM with a VTAM APPL definition, and included the correct DEVICE and LINK statements in *hlq*.PROFILE.TCPIP. For NCP V7R4, or later, IP over CDLC can be used instead of SNALINK.

If you are using the Cross Domain Resource (CDRSC), verify that the cross-domain resource managers are configured in VTAM.

Following is an example of an appropriate VTAM APPL definition:

```
*********************************************************************
*       SNALINK VTAM APPL DEFINITION                                *
*********************************************************************
SNALKLU1 APPL AUTH=(ACQ,VPACE),ACBNAME=SNALKLU1,EAS=12,PARSESS=YES,   *
              SONSCIP=YES,VPACING=0,SRBEXIT=YES
```

Note: The application name (the ACBNAME value, SNALKLU1, in this example) must match the REMLU interface definition in the NCP clients generation program. See the example in “Step 6: Configure the Host-Dependent NCP Clients” on page 214 for more information.

Following is an example of corresponding DEVICE and LINK statements:

```
;
;  DEVICE AND LINK DEFINITIONS FOR SNALINK LU0
;
DEVICE SNA1LINK SNAIUCV SNALINK A04TOLU1 SNALPROC
LINK SNALINK SAMEHOST 1 SNA1LINK
;
```

Note: The LU name on the DEVICE statement (A04TOLU1 in this example) must match the LU name of the NCST interface definition in the NCP clients generation program. See the example in "Step 6: Configure the Host-Dependent NCP Clients" on page 214 for more information.

If you want the SNALINK device to start automatically, verify that you have a START statement for this device in *hlq*.PROFILE.TCPIP. For example, `START SNA1LINK`. Otherwise, you will have to start the device manually.

Step 3: Configure the IP over CDLC DEVICE and LINK Statements

For NCPROUTE, IP over CDLC can be configured along with SNALINK for NCP V7R3, or later, or it can be used to replace SNALINK for NCP V7R4, or later.

Following is an example of corresponding DEVICE and LINK statements for the configuration shown in Figure 25 on page 211 for the MVS2 host:

```
;
;  DEVICE AND LINK DEFINITIONS FOR IP OVER CDLC
;
DEVICE IPC1NCP CDLC 013 40 40 1024 1024
LINK IPC1 CDLC 0 IPC1NCP
;
DEVICE IPC2NCP CDLC 014 40 40 1024 1024
LINK IPC2 CDLC 0 IPC2NCP
```

Note: If you want a CDLC device to start automatically, verify that you have a START statement for this device in *hlq*.PROFILE.TCPIP, for example, START IPC1NCP. Otherwise, you will have to start the device manually.

Step 4: Update the NCPROUTE Cataloged Procedure

Update the NCPROUTE cataloged procedure by copying the sample in SEZAINST(NCPROUT) to your system or recognized PROCLIB. Specify NCPROUTE parameters and change the data set names to suit your local configuration. See Figure 26 on page 219 for an illustration of NCPROUTE data set relationships.

Step 5: Update hlq.ETC.SERVICES

NCPROUTE uses the *hlq*.ETC.SERVICES data set to determine the port number on which to run. This data set can be used to define a port number other than the reserved well-known port for NCPROUTE. This data set must exist for NCPROUTE to run.

The ETC.SERVICES data set is dynamically allocated using the standard search sequence for data set names. This data set also can be explicitly allocated in the NCPROUTE cataloged procedure using the //SERVICES DD statement.

The entries in *hlq*.ETC.SERVICES are case and column sensitive. They must be in lowercase and begin in column 1.

Add the following lines to the *hlq*.ETC.SERVICES data set:

```
ncproute  580/udp
router    520/udp
```

Note: Verify that the NCPROUTE service port number is the port being used by the NCP clients. This number should match the port number defined in the NCP generation definition using the UDPPORT keyword on the IPOWNER statement. This port number does not necessarily have to match the reserved port number for NCPROUTE on the PORT statement in *hlq*.PROFILE.TCPIP.

The reserved router service port number is 520. It is required for the NCPROUTE transport of RIP packets to NCP clients which are responsible for broadcasting the packets to other RIP routers. It cannot be overridden.

If you want to use name aliases, refer to INFO APAR II08205 for information.

Step 6: Configure the Host-Dependent NCP Clients

You should refer to the appropriate NCP documentation for more information about defining and generating the NCP and creating route information tables.

- For more information about defining IP, refer to *NCP, SSP, and EP Resource Definition Guide*.
- For more information about the IP Dynamics function, refer to *NCP and EP Reference*.
- For more information about NCP generation definitions for IP, refer to *NCP, SSP, and EP Resource Definition Reference*.
- For more information about generating NCP as an IP router, refer to *NCP, SSP, and EP Generation and Loading Guide*.

Note: See NCPROUTE notes in on page 205.

Generating the Routing Information Tables: To support IP dynamics, NCP's Network Definition Facility (NDF) builds a routing information table (RIT) for networks and subnetworks for use by TCP/IP at NCP generation time.

The RIT consists of routing tables that are generated from the NCP IPROUTE and IPLOCAL statements. During NCP generation, the RIT is added as a member of the NCP load library partitioned data set ncp.v7r1.ncpload. You identify the member name of ncp.v7r1.ncpload that NCPROUTE uses at execution time with the NEWNAME parameter of the BUILD statement for each NCP client generation.

Determining the Gateway Route Table Name: There is one RIT in the ncp.v7r1.ncpload data set for each NCP client this server supports. The NCPROUTE server receives the NCP name from an NCP client in the "Hello" message. This name is used as the base to determine the member name in the ncp.v7r1.ncpload partitioned data set to use for the initial RIT for this NCP client. The RIT member name in the ncp.v7r1.ncpload data set is the NEWNAME parameter of the BUILD statement for the NCP generation with a suffix of **P** added. Specify a unique name on the NEWNAME parameter of the BUILD statement for each NCP client. This name is also used as the member name if the optional gateways data set (GATEWAYS_PDS) is specified in the NCPROUTE profile. The RIT is accessed by NCPROUTE from a //STEPLIB DD statement in the NCPROUTE cataloged procedure, LINKLST, or authorized library.

NCST Session Interface Definition: The NCP Connectionless SNA Transport (NCST) interface is used to establish a session that can provide a connection to another IP node (NCP or S/370) over a SNA network. Use this definition when

using NCST PU interfaces to communicate with NCPROUTE using SNALINK devices with the MVS host. The NCST interface must be defined to match the SNALINK LU0 interface in VTAM so that an NCP client can establish a connection with NCPROUTE. The LU statement in the NCST interface definition tells VTAM which interface to use for the SNALINK application. The following are important keywords in this definition:

NCST
Specifies the protocol type. Must be coded as IP for internet protocol.

INTFACE
Specifies the name of the interface and the maximum transfer unit (MTU) size for the NCST session to the VTAM owner (IPOWNER).

REMLU
Specifies the name of the remote LU for the SNALINK LU0 VTAM connection. This name must match:
- The APPLID in the PROC statement of the SNALINK cataloged procedure
- The application name in the VTAM APPL definition

Note: If you define a backup NCST SNALINK session, the REMLU can specify the primary logical name fo the remote LU or a different remote LU. Ensure that the MTU sizes are the same for the backup NCST sessions.

Following is an example of an NCST session interface definition:

```
*********************************************************************
*     NCST IP INTERFACES                                            *
*********************************************************************
A04NCSTG GROUP NCST=IP,LNCTL=SDLC,VIRTUAL=YES
A04NCSTL LINE LINEFVT=CXSXFVT,PUFVT=CXSXFVT,LUFVT=(CXSXFVT,CXSXFVT),   *
               LINECB=CXSXLNK
A04NCSTP PU VPACING=0,PUTYPE=2,PUCB=CXSP0000
*
A04TOLU1 LU INTFACE=(NCSTALU1,1492),REMLU=SNALKLU1,LUCB=(CXSL0000,CXSS0*
               000),LOCADDR=1
*
```

Note: The NCST LU name (A04TOLU1 in this example) must match the LU name on the SNALINK LU0 DEVICE statement in *hlq*.PROFILE.TCPIP. See the example in “Step 2: Configure VTAM and SNALINK Applications” on page 212 for more information.

Channel PU Interface Definition: Use this definition with channel PU interfaces (ESCON, BCCA, or CADS) to communicate with NCPROUTE using IP over CDLC devices with the MVS host.

Following is an example of channel PU interface definition using the ESCON channel type:

```
*********************************************************************
*   PHYSICAL ESCON CHANNEL DEFINITIONS                              *
*********************************************************************
*
A04PSOC1 GROUP LNCTL=CA,MONLINK=NO,NPACOLL=NO,XMONLNK=YES
               SPEED=144000000,SRT=(32768,32768)
*
A04S2240 LINE  ADDRESS=2240
A04P2240 PU    ANS=CONTINUE,PUTYPE=1
*
*********************************************************************
*    LOGICAL ESCON CHANNEL DEFINITIONS                              *
*********************************************************************
```

```
*
A04PSOCB GROUP LNCTL=CA,PHYSRSC=A04P2240,NPACOLL=NO,
               DELAY=0.2,MAXPU=16,MODETAB=AMODETAB,
               DLOGMOD=INTERACT,SPEED=144000000,
               SRT=(21000,20000),PUDR=YES,
               TIMEOUT=150.0,CASDL=0.0
*
A04LSOC2 LINE  ADDRESS=NONE,HOSTLINK=1
A04L2S1  PU    ADDR=01,PUTYPE=1,ARPTAB=(10,,NOTCANON),
               INTFACE=SOCPU1
A04L2S2  PU    ADDR=02,PUTYPE=1,ARPTAB=(10,,NOTCANON),
               INTFACE=SOCPU2
```

NCP Host Interface Definition: The IPOWNER statement in the NCP generation definition contains the TCP/IP host information and tells NCP which interface to use for NCPROUTE. The following are important keywords on the IPOWNER statement:

INTFACE
Specifies the name of the interface to the owning TCP/IP host that is running NCPROUTE.

HOSTADDR
Specifies the IP address of the owning TCP/IP. This address must match the IP address in the HOME statement in *hlq*.PROFILE.TCPIP data set for a SNALINK or IP over CDLC interface.

UDPPORT
Specifies the UDP port number for NCPROUTE. The default is 580. This port number must match the NCPROUTE service port number defined in the *hlq*.ETC.SERVICES data set. See “Step 5: Update hlq.ETC.SERVICES” on page 213 for more information.

The IPLOCAL statement in the NCP generation definition contains the NCP routing information for the local attached routes. During NCP generation, this information gets included in the Routing Information Table (RIT) which NCPROUTE uses to build the interface and routing tables. IPLOCAL routes are predefined as permanent or static to prevent modification by NCPROUTE. The following are important keywords on the IPLOCAL:

INTFACE
Specifies the name of the locally attached interface.

LADDR
Specifies the IP address of the locally attached interface.

P2PDEST
For point-to-point interfaces only. Specifies the IP address of the remote end of the point-to-point link.

PROTOCOL
Specifies the type of protocol to be used for the interface. The default is RIP which indicates that the interface is RIP-managed by NCPROUTE.

SNETMASK
Specifies the subnetwork mask for a route to a network that is subnetted. Because RIP does not support variable subnetwork masking, this value must be equal to the subnetwork mask of the route’s destination.

The IPROUTE statement in the NCP generation definition contains the NCP routing information for optional predefined routes. During NCP generation, this information gets included in RIT which NCPROUTE uses to add the routes to its routing tables.

IPROUTE routes can be predefined as permanent or non-permanent for route management control by NCPROUTE. The following are important keywords on the IPROUTE statement:

INTFACE
Specifies the name of the locally attached interface for the route.

DESTADDR
Specifies the route's destination IP address.

DISP
Specifies the disposition for the route. A disposition of PERM indicates that this route is a permanent route and will not be modified by NCPROUTE. The default is NONPERM.

HOSTRT
Indicates whether this is a host route. The default is NO.

NEXTADDR
Specifies the IP address of the gateway through which the route can reach its destination. A value of 0 indicates that there is no gateway.

The following example shows typical NCP RIP router generation source statements.

```
************************************************************************
*       IP ROUTING DEFINITIONS                                         *
************************************************************************
*
        IPOWNER INTFACE=NCSTALU1,HOSTADDR=9.67.116.66,                 *
                NUMROUTE=(100,100,100),MAXHELLO=25,UDPPORT=580
*
        IPLOCAL LADDR=9.67.116.65,INTFACE=NCSTALU1,METRIC=1,           *
                P2PDEST=9.67.116.66,PROTOCOL=RIP,SNETMASK=FFFFF000
        IPLOCAL LADDR=10.68.0.88,INTFACE=TR88,METRIC=1,                *
                SNETMASK=FFFFF000
        IPLOCAL LADDR=10.68.0.92,INTFACE=TR92,METRIC=1,                *
                SNETMASK=FFFFF000
*
        IPROUTE DESTADDR=11.0.0.1,NEXTADDR=0,INTFACE=TR88,METRIC=2,    *
                DISP=PERM,HOSTRT=YES
        IPROUTE DESTADDR=12.0.0.0,NEXTADDR=13.0.0.1,INTFACE=TR92,      *
                METRIC=2,DISP=NONPERM
```

The following example shows IPOWNER and IPLOCAL statements for the ESCON channel PU interfaces in the configuration for NCP2 as shown in Figure 25 on page 211.

```
************************************************************
*   IP ROUTING DEFINITIONS USING ESCON CHANNEL INTERFACES   *
************************************************************
*
IPOWNER INTFACE=SOCPU1,UDPPORT=580,NUMROUTE=(110,120,130),
        HOSTADDR=12.1.1.1
*
IPLOCAL LADDR=12.1.1.98,INTFACE=SOCPU1,METRIC=1,
        P2PDEST=12.1.1.1,PROTOCOL=RIP,SUBNETMASK=FFFFFF00
*
IPLOCAL LADDR=12.1.1.99,INTFACE=SOCPU1,METRIC=1,
        P2PDEST=12.1.1.2,PROTOCOL=RIP,SUBNETMASK=FFFFFF00
*
IPLOCAL LADDR=13.1.1.99,INTFACE=SOCPU2,METRIC=1,
        P2PDEST=13.1.1.1,PROTOCOL=RIP,SUBNETMASK=FFFFFF80
```

Step 7: Configure the NCPROUTE Profile Data Set (Optional)

To build the NCPROUTE profile, create a data set and specify its name in the //NCPRPROF DD statement in the NCPROUTE cataloged procedure. You can find

a sample in SEZAINST(EZBNRPRF). Include configuration statements in this data set to define SNMP functions and to identify the NCPROUTE gateways data set. For more information, refer to *z/OS Communications Server: IP Configuration Reference*.

RIP_SUPPLY_CONTROL *supply_control*
Specifies one of the following options on a server-wide basis:

- RIP1—Unicast/Broadcast RIP Version 1 packets (Default)
- RIP2B—Unicast/Broadcast RIP Version 2 packets (Not Recommended)
- RIP2M—Unicast/Multicast/Broadcast RIP packets (Migration)
- RIP2—Unicast/Multicast RIP Version 2 packets
- NONE—Disables sending RIP packets

Note: If RIP2 is specified, the RIP Version 2 packets are multicast over multicast-capable interfaces only. No RIP packets are sent over multicast-incapable interfaces. For RIP2M, the RIP Version 2 packets are multicast over multicast-capable interfaces and RIP Version 1 packets over multicast-incapable interfaces. For RIP2B, the RIP Version 2 packets are unicast or broadcast; this option is not recommended since host route misinterpretations by adjacent routers running RIP Version 1 can occur. For this reason, RIP2B may become obsolete in a future release. For point-to-point interfaces that are non-broadcast and multicast-incapable, the RIP Version 2 packets are unicast.

RIP_RECEIVE_CONTROL *receive_control*
Specifies one of the following options on a server-wide basis:

- RIP1—Receive RIP Version 1 packets only
- RIP2—Receive RIP Version 2 packets only
- ANY—Receive any RIP Version 1 and 2 packets (Default)
- NONE—Disables receiving RIP packets

Note: If the client NCP does not support variable subnetting, the default of ANY is changed to RIP1.

RIP2_AUTHENTICATION_KEY *authentication_key*
Specifies a plain text password *authentication_key* containing up to 16 characters. The key is used on a router-wide basis and can contain mixed case and blank characters. Single ‘ (’) can be included as delimiters to include leading and trailing blanks. The key will be used to authenticate RIP Version 2 packets and be included in the RIP updates for authentication by adjacent routers running RIP Version 2. For maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL to RIP2. This will discard RIP1 and unauthenticated RIP2 packets. A blank key indicates that authentication is disabled. Following are examples of authentication passwords:

```
my password        (no leading or trailing blanks)
' my password '    (leading and trailing blanks)
''abc''            (single quotes part of password)
'     '            (5-character blanks)
```

SNMP_AGENT *host_name*
Specifies the host name or IP address of the host running an SNMP daemon. Only one NCPROUTE server can use a particular SNMP agent at a time.

SNMP_COMMUNITY *community_name*
Specifies a community name that SNMP applications must use to access data that the agent manages. Protect this information accordingly.

GATEWAY_PDS *dsname*
Specifies the optional partitioned data set that contains GATEWAY information for each client NCP. Quotation marks are not needed when specifying *dsname*. One member for each NCP client of this data set must be configured to match the NCP NEWNAME parameter with the **P** suffix which is the same as the NCP's RIT member name. See "Step 8: Configure the NCPROUTE Gateways Data Set (Optional)" for information on defining the statements necessary for the members of this data set.

Note: You can use a semicolon in column 1 to permit comments in the profile. Blank lines are also permitted.

Figure 26 shows the relationship between the data set names specified in the NCPROUTE cataloged procedure and the NCPROUTE profile, as well as the relationship between the members of the gateways PDS and the ncpload PDS.

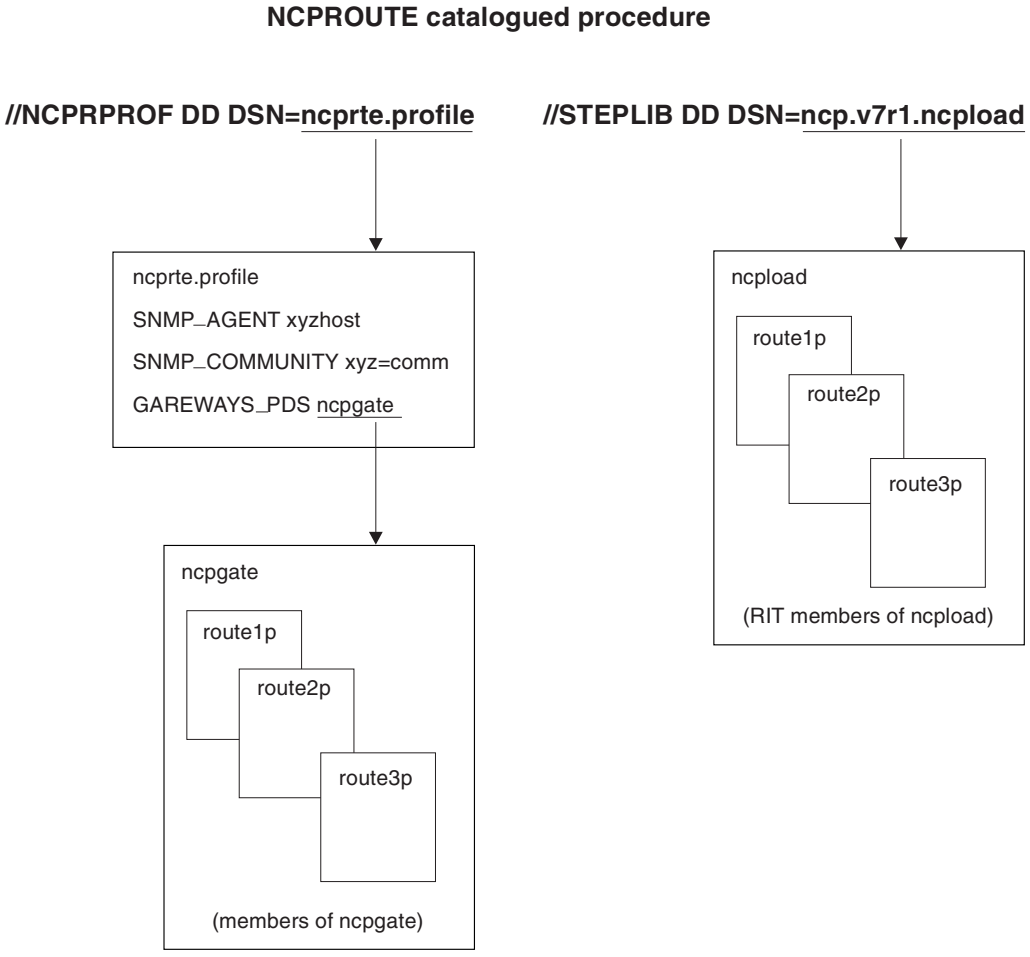


Figure 26. NCPROUTE Data Sets Relationship

Step 8: Configure the NCPROUTE Gateways Data Set (Optional)

The gateways data set is used to identify routes not defined in the NCP routing information table.

NCPROUTE and ROUTED require separate gateways data sets. The two servers cannot share the same data set. The NCPROUTE gateways data set is optional.

However, if you use it, you must include the GATEWAY_PDS statement in the NCPROUTE profile to specify the gateway data set name. The NCPROUTE server queries the gateways data set for static routing information. It also dynamically receives routing information from the NCP client portion of this RIP router.

Allocate the gateways data set with partitioned organization (PO), a fixed block format (FB), a logical record length of 80 (LRECL), and any valid block size value for a fixed block, such as 3120.

A `passive` entry in the gateways data set is used to add a route to a part of the network that does not support RIP. An `external` entry in the gateways data set indicates a route that should never be added to the routing tables. If another RIP server offers this route to your host, the route is discarded and not added to the routing tables. An `active` entry indicates a gateway that can only be reached through a network that does not allow or support link-level broadcasting or multicasting.

Note: The gateways data set is not related to the GATEWAY statement used in *hlq*.PROFILE.TCPIP data set.

To configure NCPROUTE, add an entry to the gateways data set for each route not defined in the NCP RIT. Use the options statement to define the characteristics of the routes in this member of the PDS.

Configuring a Passive Route: Figure 27 illustrates an NCPROUTE configuration using NCP as the destination hosts. In other configurations, destination hosts might not necessarily be NCPs.

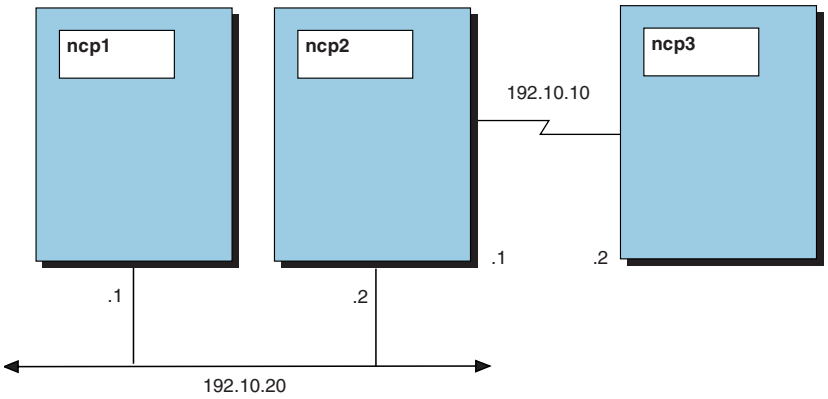


Figure 27. NCPROUTE Configuration Example of a Passive Route

Using Figure 27, assume that your NCP client `ncp1` is channel-attached to an MVS host running an NCPROUTE server. The other two NCP clients, `ncp2` and `ncp3`, are not running a RIP server. Also assume that permanent routes to `ncp2` and `ncp3` are not defined with the IPROUTE definitions in the NCP generation definition for `ncp1`. Your NCPROUTE server cannot learn a route to `ncp3`, because `ncp2` is not running a RIP server. Your NCPROUTE server sends routing updates to `ncp3` over the link to `ncp2`, but never receives a routing update from `ncp2`. After 180 seconds, your NCPROUTE server deletes the route to `ncp2`. This problem is inherent to the RIP protocol and cannot be prevented. Therefore, you need to add a `passive` route to `ncp3` in the NCPROUTE gateways data set for the NCP client `ncp1`. This is the data set defined by the GATEWAYS_PDS statement in the NCPROUTE profile.

You can use either of the following gateway statements:

```
host  ncp3         gateway  ncp2         metric 2  passive

host  192.10.10.2  gateway  192.10.20.2  metric 2  passive
```

Similarly, because ncp2 is not running an RIP server supported by NCPROUTE, you need to add a directly-connected passive route as follows:

```
host  ncp2         gateway  ncp1         metric 1  passive
```

A directly-connected passive route is one where the gateway address or name is one of the local interfaces in the NCP generation.

Assume that your NCP client is now ncp2 and is running a NCPROUTE server. ncp1 is also running a RIP server, but ncp3 is not. Your NCPROUTE server sends routing information updates to ncp3 over the link to ncp3 but never receives a routing update from ncp3. After 180 seconds, your NCPROUTE server deletes the route to ncp3.

You should add a passive route to ncp3 as follows:

```
   host  ncp3        gateway  ncp2        metric 1  passive
```

ncp1 cannot reach ncp3 unless a passive routing entry is added to ncp1. For example:

```
host   ncp3    gateway  ncp2  metric  2  passive
```

or

```
host  192.10.10.2  gateway  192.10.20.2  metric  2  passive
```

Configuring an External Route: Using Figure 27, assume that your NCP client ncp1 is channel-attached to an MVS host running an NCPROUTE server. The other two NCP clients, ncp2 and ncp3, are also running a RIP server. Your NCPROUTE server normally learns a route to ncp3 from ncp2, because ncp2 is running a RIP server. You might not want ncp1 to route to ncp3 for security reasons. For example, a university might want to prevent student hosts from routing to administrative hosts.

To prevent your NCPROUTE server from adding a route to ncp3, add an external route to the NCPROUTE gateways data set. This is the data set defined by the GATEWAYS_PDS statement in the NCPROUTE profile. You can use either of the following gateway statements:

```
host  ncp3         gateway   ncp2        metric 2  external

host  192.10.10.2  gateway  192.10.20.2  metric 2  external
```

Configuring an Active Gateway:

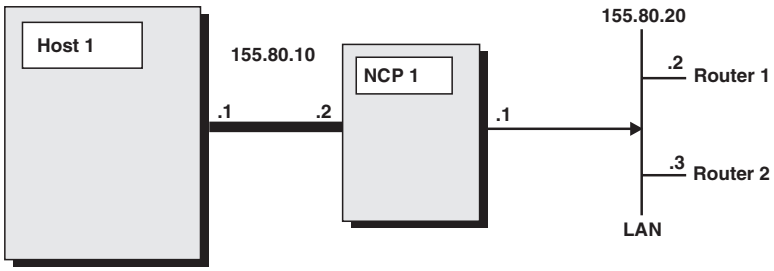


Figure 28. Configuring an Active Gateway

As shown in Figure 28, assume that your NCP client is `ncp1`, which is channel-attached to an MVS host running an NCPROUTE server and that it has a network attachment adapter that does not support link-level broadcasting or one that does not support ARP processing. Also, assume that there are routers `Router1` and `Router2` on the local area network. Because the IP addresses `Router1` and `Router2` are unknown by `ncp1`, they have to be manually configured in NCPROUTE for NCPROUTE to communicate with them. Configuring active gateways for `Router1` and `Router2` as remote network interfaces enables NCPROUTE to send RIP responses to the target addresses.

1. Specify IP addresses for each network adapter (without link-level broadcasting or ARP support) attached to the local network in the NCP client according to the NCP generation definition. For example, 155.80.20.1 is the IP address for the local network adapter attachment in `ncp1`.
2. Define active gateways for the remote routers in the NCPROUTE gateways data set specified on the GATEWAYS_PDS statement in the NCPROUTE profile:

```
active active gateway 155.80.20.2 metric 1 active
active active gateway 155.80.20.3 metric 1 active
```

NCPROUTE will use these active gateway addresses as the destination addresses to send RIP responses to the remote routers. In addition, NCPROUTE will continue to receive RIP responses from the active gateways over the NCP client.

Configuring a Default Route: A default route is typically used on a gateway or router to an internet, or on a gateway or router that uses another routing protocol, whose routes are not reported to other local gateways or routers.

To configure a route to a default destination, add a default route using the IPROUTE statement in the NCP generation definition. For example, if the default destination router has a gateway address 9.67.112.1, an IPROUTE statement might look like:

```
IPROUTE  DESTADDR=0.0.0.0,NEXTADDR=9.67.112.1,INTFACE=TR88,
         METRIC=1,DISP=PERM
```

An easier way would be to use the passive route definition specified in the NCPROUTE gateways data set for the NCP client. For example, the gateways entry would look like:

```
net 0.0.0.0 gateway 9.67.112.1 metric 1 passive
```

Only one default route to a destination gateway or router can be specified. For an NCP client, NCPROUTE currently does not support multiple default routes.

Configuration Examples: The following example shows the contents of an NCPROUTE gateways data set containing multiple entries:

```
options default.router no trace.level 4 supply on
net testnet gateway 9.0.0.100 metric 1 passive
net 2.0.0.2 gateway 9.0.0.101 metric 2 external
host 2.0.0.3 gateway 9.0.0.102 metric 3 passive
host 2.0.0.4 gateway 9.0.0.103 metric 2 external
active active gateway 2.0.1.1 metric 1 active
```

In the second entry, the route indicates that NCPROUTE can reach network `testnet` through the gateway `9.0.0.100`, and that it is one hop away. This passive route is not broadcast to other RIP routers.

In the third entry, the route indicates that NCPROUTE can reach network `2.0.0.2` through the gateway `9.0.0.101`, and that it is two hops away. Because this route is external, NCPROUTE should not add routes for this network to the routing tables and routes received from other RIP routers for this network should not be accepted.

In the fourth entry, the route indicates that NCPROUTE can reach host `2.0.0.3` through gateway `9.0.0.102`, and that it is one hop away. This passive route is not broadcast to other RIP routers.

In the fifth entry, the route indicates that NCPROUTE can reach host `2.0.0.4` through gateway `9.0.0.103`, and that it is two hops away. Because this route is external, NCPROUTE should not add routes for this network to the routing tables, and routes received from other RIP routers for this network should not be accepted.

The sixth entry shows an active gateway. Note that it is specified as the last entry in the data set.

Note: If a default route is to be defined to a destination gateway or router, configure a default route in this gateways data set (if the default route is not defined in a NCP client's generation definition).

Step 9: Define a Directly Connected Host Route for the NCST Session

If you are not using RouteD, you need to configure a directly-connected static host route using the GATEWAY statement in *hlq*.PROFILE.TCPIP. For example, if you are using SNALINK as the host route and have the IP addresses shown in Figure 25 on page 211, the GATEWAY statement might look like this:

```
GATEWAY
; net_number first_hop link_name packet_size     subnet_mask  subnet_value
  9.67.116.65     =    SNALINK     32758           HOST
```

See *z/OS Communications Server: IP Configuration Reference* and the GATEWAY syntax information in "Step 8: Configure the NCPROUTE Gateways Data Set (Optional)" on page 219 for more information about configuring GATEWAYS statements.

Note: The host routes on the MVS host are managed by TCP/IP as defined on the GATEWAY statement or by RouteD as defined on the BSDROUTINGPARMS statement. NCPROUTE does not manage the host routes on the MVS host. It only manages the routes on the NCP clients.

Controlling the NCPROUTE Address Space with the MODIFY Command

You can control most of the functions of the NCPROUTE address space from the operator's console using the MODIFY command.

For information about modifying the NCPROUTE address space with the MODIFY command, refer to *z/OS Communications Server: IP Configuration Reference*.

Chapter 6. Accessing Remote Hosts Using Telnet

Telnet is a terminal emulation protocol that allows end users to log on to remote host applications as though they were directly attached to that host. Telnet protocol requires the end user has a Telnet client emulating a type of terminal the host application will understand. The client connects to a Telnet server, which communicates with the host application. End users can generate many instances of the client on their host (usually a PC). Each client can be in session with an application on any host that has a Telnet server.

This chapter describes how to set up and use the following:

TN3270 Telnet Server
: Provides access to SNA applications on the host using Telnet TN3270E, TN3270, or linemode protocol.

z/OS UNIX Telnet server
: Provides access to z/OS UNIX shell applications on the host through Telnet protocol.

TN3270 Telnet Server

The TN3270 Telnet server is a VTAM application that activates one minor node logical unit (LU) to represent each Telnet client. These Telnet application LUs establish sessions with other VTAM host applications (for example, CICS), allowing binds with LU0 or LU2 for terminal support and LU1 or LU3 for printer support. The TN3270 Telnet server runs in the TCP/IP address space as part of TCP/IP. The Telnet server is started as part of the TCP/IP start-up procedure. To enable connections, the VTAM and TCP/IP configuration data sets must be modified with Telnet statements. These statements describe the Telnet LUs, a listening port, and the characteristics of that port. After TCP/IP is started, VARY and DISPLAY commands specifically related to the Telnet server can be used to alter the state of Telnet or display information about Telnet. For more information about these command sets, refer to *z/OS Communications Server: IP Configuration Reference*.

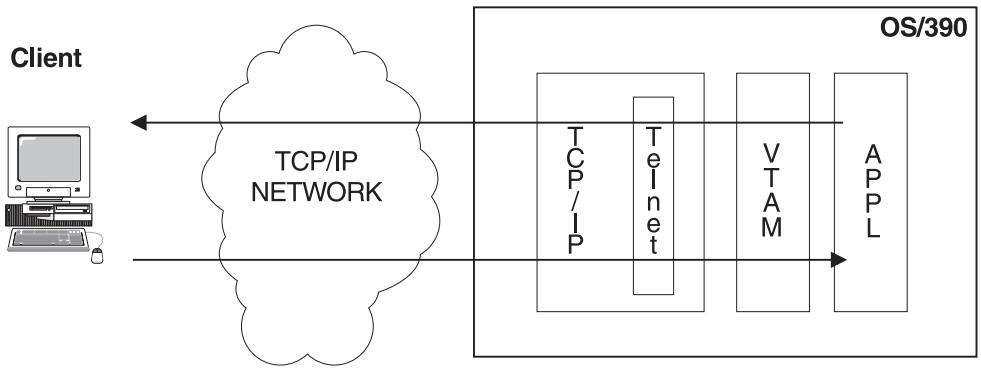


Figure 29. Telnet Connectivity

Getting Started

Telnet is part of the Initial Verification Procedure (IVP). With the IVP, an end user can:

- Start the TSO Telnet client
- Establish a Telnet connection to the Telnet server using loopback

- Begin a session with another TSO user ID on the same host

Establishing a new session confirms that the Telnet server is working properly. The VTAM configuration data set for Telnet is hlq.SEZAINST(IVPLU), and the Telnet configuration statements are part of the hlq.SEZAINST(SAMPPROF) TCP/IP configuration data set. For more information on IVP, refer to *z/OS Communications Server: IP Programmer's Reference*.

Customizing the VTAM Configuration Data Sets

Telnet uses application LUs to represent clients, and their definition members must be available to VTAM when it is started. The IVP VTAM configuration data set for Telnet is hlq.SEZAINST(IVPLU), and it must be in the concatenation of data sets specified on the VTAMLST DD statement in the procedure used to start VTAM. This member contains the Telnet application LUs, and ensures that these LUs will be available to activate after VTAM is started. To automatically activate the application definition deck, include it in ATCCONxx. For example, if multiple TCP/IP stacks with multiple Telnet servers are used in a sysplex distributor environment, ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will fail the OPEN ACB request.

Telnet LUs, representing either terminal or printer emulators, can be defined to the VTAM configuration data set using a wildcard character instead of coding individual Telnet application LU statements. The Model Application Names function allows system administrators to code a generic APPL name with an asterisk (*). Use * as a wildcard character to replace a character string at the same position anywhere within the minor node name. This can produce a significant administration savings. For example, assume 20,000 Telnet LUs are needed in the range of TCP00001 to TCP20000. A single VTAM application definition statement with a Telnet application minor node (Telnet LU) name of TCP* can support all 20,000 LUs.

Because Telnet server LUs cannot manage multiple concurrent sessions, VTAM will automatically set SESSLIM=YES for Telnet LUs defined to VTAM.

If EAS is allowed to default, the common service area (CSA) storage can be excessive. To avoid this, code EAS=1 for Telnet LUs defined to VTAM.

Customizing the TCP/IP Configuration Data Sets

Telnet can process Telnet configuration statements at any time using the OBEYFILE command or during initialization of the TCP/IP address space. Use a separate profile member with only Telnet statements (TELNETPARMS and BEGINVTAM blocks) to keep TCP/IP configuration more organized and allow for easy Telnet updates (with OBEYFILE).

To validate a Telnet profile without using the profile, specify TESTMODE (a TELNETPARMS statement). When no errors are reported, remove the TESTMODE statement and use the INCLUDE statement to add it to the TCP/IP start-up profile. The INCLUDE file isolates the Telnet statements for easier OBEYFILE processing. It is recommended, but not required, to reserve the port or ports, for Telnet by using the stand-alone PORT xx INTCLIEN statement in the TCP/IP start-up profile. If you do not code the PORT xx INTCLIEN statement, another application might use the port before the Telnet application can use it.

The hlq.SEZAINST(SAMPPROF) TCP/IP configuration data set contains sample Telnet statements that will work with the sample VTAM configuration data set described above.

The DEFAULTLUS statement defines a range of LUs. This simplifies coding. The sample profile allows Telnet to use application LUs TCPACB01 through TCPACB99 to represent clients connecting to port 23 of the TCP/IP IP address. If multiple TCP/IP stacks with multiple Telnet servers are used in a sysplex distributor environment, ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will fail the OPEN ACB request. After connecting, the DEFAULTAPPL statement causes the Telnet LU to establish a session with TSO immediately. If an error occurs during session initiation, the MSG07 statement sends an error message to the client. If MSG07 is not coded, the connection is dropped. The sample profile contains additional statements that are included as comments. These statements provide examples of how to specify advanced functions. Many of these statements are installation-specific and will require modification.

Telnet configuration uses the following statement blocks:

TELNETGLOBALS/ENDTELNETGLOBALS
: An **optional** statement block. The statements apply across multiple ports. Refer to *z/OS Communications Server: IP Configuration Reference* for more detailed information.

TELNETPARMS/ENDTELNETPARMS
: A **required** statement block. The statements define port characteristics such as port number, timers, and connection characteristics. Refer to *z/OS Communications Server: IP Configuration Reference* for more detailed information.

BEGINVTAM/ENDVTAM
: A **required** statement block. The statements define LU names to be used, client access, and how LU names and applications are assigned to clients. Refer to *z/OS Communications Server: IP Configuration Reference* for more detailed information.

Starting Telnet

You can use the IVP sample configuration data sets to start a simple Telnet. Follow these steps to establish a TSO session over a Telnet connection:

1. Ensure the sample hlq.SEZAINST(IVPLU) member is in the concatenation of data sets specified on the VTAMLST DD statement in the procedure used to start VTAM. The LUs need to be in a connectable state. To do this, activate the MAJOR NODE.
2. When the TCP/IP stack is started using hlq.SEZAINST(SAMPPROF), message EZZ6003I is displayed. This indicates that Telnet is ready to accept connections.
3. To start a TSO Telnet client emulator, specify the TCP/IP IP loopback address and the Telnet port. A TSO session will be established with the first Telnet LU (TCPABC01).
4. Enter any valid TSO user ID.

This sample is designed to verify that the basic Telnet requirements are in place and functioning. Several features are not included in this sample, but you should consider them before using Telnet in production. For example, security, LU and application assignment, and level of connection persistence are some Telnet features. To learn more about Telnet capabilities and better understand TN3270, read the remainder of this chapter.

Managing the Telnet Server

Commands

Many networking products (such as VTAM) use VARY commands to change the state of a device and DISPLAY commands to show information. The Telnet server uses VARY and DISPLAY commands to help monitor Telnet function and debug problems.

The following commands help manage the Telnet server:

- Telnet VARY commands allow the operator to stop and start Telnet and enable clients connections. These commands include:
  - QUIESCE the port to block any new connections from starting but allow existing connections to continue activity
  - RESUME to end the QUIESCEd state
  - STOP Telnet to end connections to a Telnet port and close the port
  - Start or restart a port using the VARY OBEYFILE command (to update the Telnet PROFILE). Use this command to stop Telnet activity on one port and begin activity on a new port without stopping the TCP/IP stack. You can also start activity on new ports without stopping activity on existing ports.
  - ACTivate and INACTivate LUs from the Telnet server's perspective. If an LU is already in use, the INACT command fails. Specify the name ALL to activate all inactive LUs with one command. These commands have no effect on the VTAM state of the LU.

  WLM is affected by port changes. See "WorkLoad Manager for Telnet (WLM)" on page 230 for more information.
- Telnet DISPLAY commands are discussed in "Telnet Diagnostics" on page 291.

Complete Profile Replacement

On a per-port basis, new profile statements completely replace the profile statements that were in use before the update. The updates are *not* cumulative from the previous profile. If only one change is needed in the new profile, update the old profile with the change or copy it to another data set member and update. After processing, the new profile is labeled the CURRent profile, and the replaced profile becomes profile 0001. If another update is performed and the new update becomes the current profile, the replaced profile becomes profile 0002. Use the VARY OBEYFILE command to replace a profile. If the profile statements for a VARY OBEYFILE contain a subset of the active ports, the remaining ports are unchanged. The structural layout of the profiles and connections is show in the following figure.

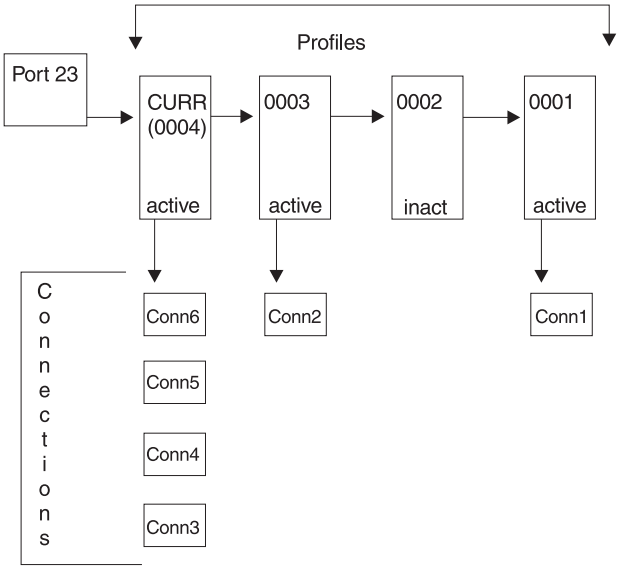


Figure 30. Telnet Profiles and Connections

Connection Association

Any new connection will use the tables generated by the most recent CURRent profile. The tables generated by older profiles remain active and continue to support any connections that were established when the older profile was the CURRent profile. When all connections associated with an older profile have ended, the storage for the older profile tables is freed and the profile is considered INACTIVE. This is permitted because all new connections must use the CURRent profile.

Multiple Ports

Telnet supports up to 255 ports on one TCP/IP stack. A unique TELNETPARMS block must be created for each port. Telnet allows the use of the same BEGINVTAM block for all ports, some ports, or a unique BEGINVTAM block for each port. There are several reasons more than one Telnet port might be needed. The two most common reasons are discussed in the following sections.

Assigning a single application to a port simplifies the setup of clients on the workstation and the logon process. Workstation clients can be labeled with the associated application name and then be set up to connect to the appropriate port. With a client per application on the workstation, the end user can select the needed client, connect, and be immediately in session with the application defined on the DEFAULTAPPL statement in BEGINVTAM. This implementation requires a unique BEGINVTAM block for each port due to the unique DEFAULTAPPL statements. The example below shows how to set up TSO, IMS, and CICS on ports 23, 223, and 423, respectively. The same LU names are used in each BEGINVTAM block. Telnet maintains a master LU ″in-use″ registry across all ports so that the same LU name will not be used by two different ports.

```
TELNETPARMS
      PORT  23
ENDTELNETPARMS
TELNETPARMS
      PORT  223
ENDTELNETPARMS
TELNETPARMS
      PORT  423
ENDTELNETPARMS
```

```
BEGINVTAM
      PORT  23
      DEFAULTLUS
        LU001..LU999
      ENDDEFAULTLUS
      DEFAULTAPPL   TSO
ENDVTAM
BEGINVTAM
      PORT  223
      DEFAULTLUS
         LU001..LU999
      ENDDEFAULTLUS
      DEFAULTAPPL   IMS
ENDVTAM
BEGINVTAM
      PORT  423
      DEFAULTLUS
         LU001..LU999
      ENDDEFAULTLUS
      DEFAULTAPPL   CICS
ENDVTAM
```

Assigning different security levels to different ports is an easy way to differentiate client security needs. External connections might require SSL security, while internal connections do not. Other than that difference, all other aspects of the Telnet profile can be the same. For example, external clients can connect to port 23 of a firewall that converts the request to the Telnet secure port 992. Internal clients would connect directly to the Telnet basic port 23. The statements below show how two ports allow implementation of different security levels. Note the same BEGINVTAM block is used for both ports, which can significantly reduce profile maintenance complexity. The PORT statement in BEGINVTAM links the BEGINVTAM block to the multiple TELNETPARMS blocks defined.

```
TELNETPARMS
      PORT  23
ENDTELNETPARMS
TELNETPARMS
      SECUREPORT  992
      KEYRING hfs /use/keyring/tcpcs.kdb
ENDTELNETPARMS
BEGINVTAM
      PORT  23 992
      DEFAULTLUS
        LU001..LU999
      ENDDEFAULTLUS
      ALLOWAPPL   *
ENDVTAM
```

If a profile that contains a new port number is processed, it is treated as an additional port, and the VARY OBEYFILE request will succeed if all parameters are correctly specified. Existing, non-referenced ports remain active and unchanged. You can use the VARY TELNET,STOP command to stop a port.

See “WorkLoad Manager for Telnet (WLM)” for more information about multiport considerations.

WorkLoad Manager for Telnet (WLM)

Telnet can be part of a large sysplex environment where the server is replicated on many machines. One of the goals of such a system is to balance workload across the different machines. WLM is a method used to direct connection requests to various machines within the sysplex. See “Chapter 8. Domain Name System (DNS)” on page 329 for more details about WLM. The WLMCLUSTERNAME statement in

TELNETPARMS is used by Telnet to specify WLM registration names. For example, assume Telnet resides on three machines.

- Machine 1 supports CICS and TSO.
- Machine 2 supports CICS, TSO, and IMS.
- Machine 3 supports CICS and IMS.

The TELNETPARMS statements needed to support this example are shown below. The registration names have been changed to avoid confusion with the application name by adding a 'TN' to the WLM registration name.

```
TELNETPARMS     ; Machine 1
  PORT  23
  WLMCLUSTERNAME  TNCICS TNTSO   ENDWLMCLUSTERNAME
ENDTELNETPARMS
TELNETPARMS     ; Machine 2
  PORT  23
  WLMCLUSTERNAME  TNCICS TNTSO TNIMS  ENDWLMCLUSTERNAME
ENDTELNETPARMS
TELNETPARMS     ; Machine 3
  PORT  23
  WLMCLUSTERNAME  TNCICS TNIMS   ENDWLMCLUSTERNAME
ENDTELNETPARMS
```

WLM algorithms are used to determine how to balance the CICS, TSO, and IMS workload. For example, a client connecting to CICS would be directed to Machine 1, 2, or 3 depending on the number of pre-existing connections and the algorithm used to manage each machine's workload. When the Telnet server is activated, all names within the WLMCLUSTERNAME statement are registered with the WLM server. If the Telnet port is quiesced or stopped, Telnet deregisters these names from WLM so that no new connection requests will be received. When the port is resumed, the names will be registered again. To deregister only one name from a port, remove the name from the profile and issue a VARY OBEYFILE command.

Assigning the same WLM cluster name to multiple ports on one TCP/IP stack is permitted. To accommodate multiple ports registering with the same WLM cluster name, only the first port registers the name. If additional ports are activated with the same WLM name, Telnet keeps track of the new ports using the name, but it does not register the name again. A WLM name will be deregistered only when the last active port using that name is being quiesced or stopped. This option works well if all machines have the same Telnet ports defined and the ports remain active. Based on the example above, assume that ports 23 and 623 are defined on Machines 1, 2, and 3. A client choosing either port will still be routed to Machine 1, 2, or 3 based on the WLM algorithm. However, if Machine 3 loses port 623, the name remains registered on Machine 3 because port 23 is still active. WLM continues to send Requests for port 623 to Machine 3, but these requests will fail.

Connection Mode

The negotiation process is designed hierarchically; TN3270E is the highest level, and linemode is the lowest.

The TN3270 Telnet server supports the following types of connections:

- TN3270 Enhanced (TN3270E)
- TN3270
- Linemode
  - Standard
  - Binary

– Transform

TN3270E is the default connection mode for the TN3270 Telnet server. If the client refuses TN3270E mode, the server tries TN3270 mode. If the client refuses TN3270 mode, the server then tries Linemode.

Note: Within Linemode, Transform and Binary are special-case connection types.

The client might refuse a connection mode for the following reasons:

- It does not support the specified mode.
- It is set up to use a specific mode.

TN3270E and TN3270 are very similar. If the TN3270E functions described in the following sections are not needed, the end user does not notice any difference between TN3270E and TN3270 connections. In some cases, older clients do not properly refuse the server request for a TN3270E connection, and the connection is dropped. In these unusual cases, code the NOTN3270E statement in TELNETPARMS to disable the TN3270E function at the server.

The ATTN key function is supported over TN3270, TN3270E, and Transform Linemode connections. It is not supported over Standard or Binary Linemode. Default LOGMODES for TN3270E connections are SNA, and default LOGMODES for TN3270 and Transform connections are non-SNA. Telnet processes the ATTN key differently for SNA and non-SNA LOGMODES. See “Device Types and Logmode Considerations” on page 252 for more information.

TN3270 Enhanced (TN3270E)

TN3270E connections support full-screen 3270 emulation that is sometimes referred to as TN3270 Extended. Do not confuse TN3270E function with the IBM 327x device types that end in -E (for example, 3278-2-E). In these cases, the *E* indicates that the terminal supports Extended field attributes such as color and highlighting, but it is not related to Telnet functions.

Telnet is often used as the primary method of connection between client workstations and the SNA mainframe environment. To make this form of remote connection as seamless as possible, Telnet terminal emulation should simulate actual SNA terminals as closely as possible.

To accomplish this, RFC1647 and RFC2355 (both known as TN3270E) add the ability to specify device names at connection time, add support for printer devices, and add additional SNA functions.

Device name specification

The Telnet server uses an LU assignment algorithm created from the supplied configuration statements. Clients are assigned a device name (Telnet LU name) based on that algorithm. However, a TN3270E client can optionally request that a particular device name be assigned, or it might request that a device name from a specified pool of LUs be assigned. If the device name is allowed for this client (based on the LU assignment algorithm) and it is available, the server assigns the requested device name or a device name from the requested pool of LUs. Otherwise, the request is rejected with an appropriate reason code, and the connection is dropped. See “LU Assignment– Objects, Client Identifiers, Mapping Statements” on page 235 for additional LU assignment information.

328x printer support

Many Telnet clients emulate 328x class printers (device type IBM-3287-1).

Most support both SNA Character Stream (SCS) as an LU1 and 3270 data stream as an LU3. The support of each is negotiated at connection time. The bind initiating each session is sent to the client, and it informs the emulator which data stream to expect. When connected in TN3270E mode, the Telnet server supports these emulators in a manner similar to terminal LUs. The VTAM application perceives that the Telnet LU is an actual 3287-class printer and sends the SCS or 3270 data to the Telnet LU. Telnet passes the data on to the client, which prints the data. Telnet printer support allows you to use a single product, Telnet, to control both SNA terminals and SNA printers.

Some clients also allow a printer to be associated with a terminal device name. Using printer association, end users can connect their Telnet terminals to an application and then have Telnet assign a printer device name based on the terminal name given in the printer connection request. To associate printers with terminals, Telnet must have a printer device pool of LUs defined and a terminal device pool of LUs defined with each having the same number of device names. These two pools are associated with the LUMAP statement in BEGINVTAM. For example, a CICS table might specify that if terminal LU1 is requesting printer function, the output should be routed to printer PRT1 that is associated with terminal LU1. After the same terminal-to-printer association (LU1-PRT1) has been set up in the Telnet profile, Telnet automatically assigns the correct printer name. When the printer session is started at the client, it uses an associated connection request that specifys terminal LU1. The Telnet server ensures that the associated printer device (PRT1) is used. See “Printers” on page 242 for detailed examples of printer association.

Additional Negotiated 3270 support

The following functions are supported by most clients that support TN3270E connections:

- Responses - The client or host application can request that it receive and provide definite, exception, or no response. The client will respond to these requests and provide more accurate response information than if a TN3270 connection mode is used. For TN3270 connections, the server must intercept response requests from the host and respond on behalf of the client, incorrectly reducing measured response time.
- SysReq function - The end user can request that the connection be dropped by entering LOGOFF (in upper, lower, or mixed case) after pressing the SysReq key. Otherwise, if the application supports LUSTAT presentation screen is lost, pressing the SysReq key a second time refreshes the previous screen to the client emulator.

TN3270

TN3270 connections support full-screen 3270 emulation. TN3270 connections do not:

- Support device name specification
- Support printers
- Transmit response requests between the server and client

RFC1646 defines device name specification and printer support for TN3270 connections. However, this RFC is not supported on the TN3270 Telnet server. If either of these requests is received on a TN3270 connection, the server will drop the connection.

Linemode

In some cases, the client or the application does not support full-screen presentation. In other cases, the end user needs to work in a linemode environment. For these reasons, most emulators support linemode. Linemode supports a *go-ahead* function to simulate a half-duplex format. With *go-ahead* negotiated, the partner cannot send data until it receives a *go-ahead* from the current sender of data. In most cases, sessions are naturally half-duplex and the *go-ahead* adds unneeded transmissions. Therefore, the default is to Suppress Go Ahead (SGA). If *go-ahead* is needed to maintain a half-duplex format, code the DISABLESGA statement in TELNETPARMS.

The following are types of Linemode connections.

- Standard
- Binary
- Transform

These are described in the following sections.

Standard Linemode is assumed if no other linemode statements are specified in TELNETPARMS or the device type is not supported by one of the transform methods. Standard Linemode is the only connection mode that requires translation by Telnet. Telnet supports NLS for standard Linemode connections. ASCII and EBCDIC code pages are the basis for translation. Telnet makes use of the National Language Support ICONV services available in the C runtime library. For custom code page information, refer to the ICONV services in *z/OS C/C++ Programming Guide*. When ASCII and EBCDIC code pages are specified, a conversion descriptor will be given to Telnet. Telnet creates ASCII-EBCDIC and EBCDIC-ASCII translation tables based on the conversion descriptor. The CODEPAGE statement in TELNETPARMS is used to specify the code page names. The possible results from CODEPAGE processing are:

- If a conversion descriptor is not returned, CODEPAGE is not coded, or there is an error in the syntax, a default code page of ISO8859-1 will be used for ASCII, and the language environment code page taken from locale information will be used as the EBCDIC code page.
- If a conversion descriptor is not returned again, a default code page of IBM-1047 will be used for EBCDIC.
- If a conversion descriptor is not returned again, hardcoded translation tables within Telnet will be used. They match what would be generated if ISO8859-1 and IBM-1047 had worked.

No message is issued to the console if the first conversion succeeds. If there is any conversion failure a message is issued. If one of the later conversions succeeds, a message is issued indicating success.

Binary Linemode is set by the BINARYLINEMODE statement in TELNETPARMS. It indicates that Telnet should not do translation. The ASCII data from the client should be passed as-is to the VTAM application.

Transform Linemode is set by the DBCSTRANSFORM statement in TELNETPARMS. When coded, all data that passes through Telnet will be *transformed* from DBCS or SBCS ASCII full screen to 3270 full screen for all supported device types. If the device type is not supported, Standard or Binary Linemode is used.

Note: Transform can only be used by one port if multiple ports are active on one TCP/IP stack.

DBCSTRANSFORM can be used for either the VT100 single-byte character set (SBCS) or VT282 double-byte character set (DBCS) transform mode. When this option is chosen and the TCP/IP procedure JCL has been modified, as shown below, ASCII-based terminal emulators (VT100 or VT282) will appear as full-screen 3270 terminals. The Telnet server receives ASCII data from the client and transforms it into SBCS or DBCS EBCDIC data, depending on the terminal type. Telnet adds appropriate SNA control bytes to give the appearance that the data is coming from a 3270 terminal. The Telnet server receives EBCDIC data from the host application and transforms the SNA control bytes and data into appropriate ASCII control bytes and data. The data is sent to the ASCII-based terminal where it is displayed in 3270 full-screen emulation. DBCSTRANSFORM requires additional special Data Definition (DD) statements in the TCP/IP procedure and the DBCSTRANSFORM statement in TELNETPARMS.

You must add the three following DD statements to the TCP/IP procedure JCL to support Transform:

```
//TNDBCSCN DD DSN=hlq.SEZAINST(TNDBCSCN),DISP=SHR
//TNDBCSXL DD DSN=hlq.SEZAXLD2,DISP=SHR
//TNDBCSER DD SYSOUT=*
```

- The TNDBCSCN DD statement must point to the configuration data set for 3270 DBCS transform mode. This configuration data set specifies the default DBCS conversion mode that will take effect at initialization time. Specify the CODEKIND and CHARMODE parameters according to the required DBCS code page. If CODEKIND and CHARMODE are not specified, or if the TNDBCSCN DD statement is not configured, CODEKIND defaults to SJISKANJI and CHARMODE defaults to ALPHABET. A sample can be found in hlq.SEZAINST(TNDBCSCN).
- The TNDBCSXL DD statement must point to the data set containing binary translation table codefiles for 3270 DBCS transform mode. The installation data set, hlq.SEZAXLD2, contains the default binary translation table codefiles. The binary translation table codefiles for 3270 Transform can be customized by using the CONVXLAT command. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about customizing translation table code files.
- The TNDBCSER DD statement defines where Transform-specific error messages are recorded. This DD statement can specify an output data set or SYSOUT=*.

Specifying the DBCSTRACE statement in TELNETPARMS sends detailed trace output from 3270 Transform to the location specified in the SYSPRINT output DD statement. Additional detailed trace output is also sent to TNDBCSER. Both data sets will contain detailed trace data.

LU Assignment– Objects, Client Identifiers, Mapping Statements

The TN3270 Telnet server provides flexibility for assigning objects to clients based on Client Identifiers. This section provides definitions, rules, and examples of many assignment methods. Examples start with simple concepts, then progress to more complicated concepts showing interaction between assignment statements. All statements relating to assignments are listed below and are in the BEGINVTAM block, except NOTN3270E and SIMCLIENTLU, which are in the TELNETPARMS block. Refer to *z/OS Communications Server: IP Configuration Reference* for statement rules not discussed here.

The general relationship of assignment statements is:

MAP the **OBJECT** to the client based on **CLIENT IDENTIFIER**

Objects that can be assigned to clients include the following:

- LU name - The exact LU name that represents a terminal or printer emulator.
- LUGROUP - A group of terminal LU names. LUGROUP is a Telnet statement.
- PRTGROUP - A group of printer LU names. PRTGROUP is a Telnet statement.
- DEFAULTLUS or DEFAULTLUSSPEC - A default group of terminal LU names. DEFAULTLUS and DEFAULTLUSSPEC are Telnet statements.
- DEFAULTPRT or DEFAULTPRTSPEC - A default group of printer LU names. DEFAULTPRT and DEFAULTPRTSPEC are Telnet statements.
- Application name - The host application name used by the client connection.
- USS Table - The USS table name used by the client connection.
- INTERPRET Table - The INTERPRET table name used by the client connection.
- PARMSGROUP - A group of parameters that help define the connection. PARMSGROUP is a Telnet statement.

Client Identifiers are based on IP address, host name, or link name. The following identifiers can be used on most mapping statements.

- IP address - The exact source IP address of a client.
- IPGROUP name - A group of IP addresses. IPGROUP is a Telnet statement.
- Host name - The exact host name of the client.
- HNGROUP name - A group of host names. HNGROUP is a Telnet statement.
- Link name - The TCP/IP Link name identified by the client destination IP address.

Mapping statements map an Object to a client based on Client Identifier. Some Objects are actual names and others are groups of names created by Telnet statements. For example, LUGROUP is a Telnet statement that defines a group of LU names.

- LUMAP - Maps a terminal LU name or group of names (LUGROUP) to a Client Identifier.
- PRTMAP - Maps a printer LU name or group of names (PRTGROUP) to a Client Identifier.
- DEFAULTAPPL - Maps an applname to a Client Identifier for connection modes other than linemode.
- LINEMODEAPPL - Maps an applname to a Client Identifier for a connection mode of linemode.
- USSTCP - Maps a USS table to a Client Identifier. See “Using the Telnet Solicitor or USS Logon Panel” on page 279 for more USS table information.
- INTERPTCP - Maps an Interpret table to a Client Identifier. See “Using the Telnet Solicitor or USS Logon Panel” on page 279 for more Interpret table information.
- PARMSMAP - Maps additional parameters (PARMSGROUP) to a Client Identifier.

The following are additional statements relating to LU assignments.

- ALLOWAPPL - Supports LU name or LUGROUP name mapping based on application name. See “Application Security” on page 253 for more ALLOWAPPL information.
- RESTRICTAPPL - Supports LU name or LUGROUP name mapping based on user ID. See “Application Security” on page 253 for more RESTRICTAPPL information.

- NOTN3270E - Turns off TN3270E connection mode which can affect assignment outcome.
- SIMCLIENTLU - For TN3270E connections, postpones LU assignment until the application name is known.

Objects

When a client connection request is made, Telnet must assign an LU name to represent the client. Optionally, a USS table, default application, or unique parameters defined in the PARMSGROUP statement can be assigned to the connection. The LU name, tables, and parameter group are all objects that are assigned to the connection. See “Mapping Statements” on page 238 for details about how these objects are mapped to clients.

Client Identifiers

Each Telnet mapping statement can use a different Client Identifier. In fact, one client can be represented by many different Client Identifiers. For example, Telnet might assign an LU based on client host name, assign an application based on a client IPGROUP name, and assign a USS table based on link name. In some cases, two different Client Identifiers that represent the same client are used on mapping statements to map the same type of Object. In these cases, Telnet must determine which Client Identifier to use when assigning the Object. See “Client Identifier Selection Rules” on page 249 for more details.

The following are things you should consider when deciding which Client Identifiers to use.

- IP address or IPGROUP name - Client source IP address is the most common method used to map objects to the client. In a static network, Objects can be exactly mapped to clients based on the IP address, or Objects can be mapped to IPGROUP names containing exact IP addresses and subnets. For example, LUADMN is mapped to exact IP address 1.1.1.1, and application PAYROLL is mapped to IPGROUP name IPGPAY.

```
IPGROUP IPGPAY
   1.1.2.2  1.1.2.3
   255.255.0.0:2.2.0.0
ENDIPGROUP
LUMAP LUADMN 1.1.1.1
DEFAULTAPPL  PAYROLL IPGPAY
```

- Host name or HNGROUP name - If the network dynamically assigns IP addresses, the same client will not have the same IP address from one connection to the next. However, if Dynamic Domain Name System (DDNS) and Dynamic Host Configuration Protocol (DHCP) are used, the client host name can be constant. See “Chapter 8. Domain Name System (DNS)” on page 329 for more DDNS and DHCP information. For static host names, Objects can be exactly mapped to clients based on their host name, or Objects can be mapped to HNGROUP names containing exact host names and wildcarded host names. For example, LUADMNM is mapped to exact host name ADMIN.DEPT1.GROUP1.COM, and application INVENTRY is mapped to HNGROUP name HNGINV.

```
HNGROUP HNGINV
   INV1.DEPT1.GROUP1.COM
   *.DEPT3.GROUP1.COM
   **.GROUP3.COM
ENDHNGROUP
LUMAP LUADMNM  ADMIN.DEPT1.GROUP1.COM
DEFAULTAPPL    INVENTRY  HNGINV
```

Telnet uses the resolver defined in the SYSTCPD file and does not look at the z/OS environmental variable (Resolver_Config) JCL statement. Host name specification requires that Telnet resolve a host name from an IP address by using the TCP/IP Resolver. To do this, a valid TCPIP.DATA data set must be provided. See “Chapter 1. Configuration Overview” on page 3 for a description of how TCPIP.DATA is located. Neither the Resolver_Config nor the /etc/resolv.conf HFS will be used when searching for TCPIP.DATA. The most common reason for message EZZ6011I, INIT_API failure, is that no resolver has been defined.

- Linkname - A linkname is defined by the TCP/IP statements DEVICE and LINK. The linkname defines a host IP address that is a destination address for clients connecting from their source IP address. Linkname is used less frequently than IP address but can be useful in cases where Object assignment is dependent on the client DESTINATION IP address instead of the client SOURCE IP address. By using a link name on mapping statements, a client can connect to LINK1 IP address to get one set of Objects or connect to LINK2 IP address to get a different set of Objects. For example, by connecting to LINK1, a client will be assigned an LU from LUGROUP name LUGLNK1 and will establish a session with TPX1. By connecting to LINK2, a client will be assigned an LU from LUGROUP name LUGLNK2 and will establish a session with TPX2.

```
LUMAP        LUGLNK1  LINK1
DEFAULTAPPL  TPX1     LINK1
LUMAP        LUGLNK2  LINK2
DEFAULTAPPL  TPX2     LINK2
```

Mapping Statements

Common Mapping Methods: Client Identifiers are known shortly after a connection request begins. In most cases, Client Identifiers are used to map Objects to the client. You can use the statements and examples shown below to create many different assignment algorithms for terminal LU emulation. More complex topics are discussed in following sections.

- DEFAULTLUS
- ALLOWAPPL
- LUMAP
- LUGROUP
- IPGROUP
- HNGROUP
- DEFAULTAPPL
- LINEMODEAPPL
- USSTCP
- INTERPTCP

The most simple LU assignment is to create an LU pool that all connections will use by default. The DEFAULTLUS statement defines this pool. The most simple application access is to allow a session with any host application. The ALLOWAPPL statement defines which application sessions are allowed. The statements shown below will cause Telnet to assign one of the 999 LUDFxx and LUDFxxx LUs to represent a client connecting from any IP address. Telnet will send a solicitor screen to the client prompting for an application name. In this example, any application name specified will be allowed by the server. Note the use of LU range specification and application name wildcarding.

```
DEFAULTLUS
  LUDF01  LUDF02  LUDF03
  LUDF04..LUDF99
  LUDF100..LUDF999
ENDDEFAULTLUS
ALLOWAPPL *
```

Assume a system administrator always connects from IP address 1.1.1.1 and always wants to be represented as LUADMN because this LU has special privileges. Also, another system administrator wants to connect using dynamic IP addressing from a mobile host named ADMIN.DEPT1.GROUP1.COM and always wants to be represented as LUADMNM (LUADMN Mobile). The LUMAP statements shown below will cause Telnet to assign LUADMN to represent the client connecting from IP address 1.1.1.1 and assign LUADMNM to represent the client connecting from host name ADMIN.DEPT1.GROUP1.COM. The LUMAP statement has priority over the DEFAULTLUS pool. If a client is mapped by the LUMAP statement, the DEFAULTLUS pool is not checked. See “LU Mapping Selection Rules” on page 250 for exact rules.

```
LUMAP LUADMN  1.1.1.1
LUMAP LUADMNM ADMIN.DEPT1.GROUP1.COM
```

If multiple LUs need to access a payroll program, use an LUGROUP statement to define the LU pool. In the example below, 20 LUs are defined in LUGPAY. The LUMAP statement causes Telnet to assign one of 20 LUs to represent a client connecting from IP address 1.1.2.1. Single LU designations and an LU range are used to show both formats.

```
LUGROUP  LUGPAY
    LUPAY01,LUPAY02,LUPAY03,LUPAY04,LUPAY05
LUPAY06.........LUPAY20
ENDLUGROUP
LUMAP LUGPAY 1.1.2.1
```

Assume payroll employees use clients from other IP addresses. They can also use the LUGPAY pool of LUs by creating an IPGROUP statement that defines the IP addresses. The example shown below defines exact IP addresses for managers and a subnet definition for the remaining employees. Any IP address that begins with 2.2 will match the subnet definition below. The LUMAP statement causes Telnet to assign one of 20 LUs to represent a client connecting from any of the IP addresses defined in IPGPAY.

```
IPGROUP IPGPAY
   1.1.2.2  1.1.2.3
   255.255.0.0:2.2.0.0
ENDIPGROUP
LUMAP LUGPAY IPGPAY
```

Assume inventory employees use clients with dynamic IP addresses but have constant host names and need certain LU names to access the inventory application. Use the HNGROUP statement to define the host names and the LUMAP statement to map the LUs to this Client Identifier. Note the use of wildcards. Any client with a host name ending in DEPT3.GROUP1.COM with only one more qualifier to the left is a match. Any client with a host name ending in GROUP3.COM with any number of qualifiers to the left is a match.

```
LUGROUP  LUGINV1
     LUINV01..LUINV20
ENDLUGROUP
HNGROUP  HNGINV
     INV1.DEPT1.GROUP1.COM
```

```
     *.DEPT3.GROUP1.COM
     **.GROUP3.COM
ENDHNGROUP
LUMAP LUGINV1  HNGINV
```

Instead of Telnet soliciting an application name from the end user, you might want Telnet to assign a default application to the client based on the Client Identifier. DEFAULTAPPL does this for connection modes other than linemode. LINEMODEAPPL does this for a connection mode of linemode. See “Application Security” on page 253 for use and parameter information. In other cases, a USSMSG10 screen might be the desired first screen the end user sees. USS tables assignments are based on the Client Identifiers using the USSTCP statement. In some cases, the USSCMD from the end user needs to be processed by an interpret table instead of the USS table. Interpret tables assignments are based on the Client Identifier using the INTERPTCP statement. See “Using the Telnet USS and INTERPRET Support” on page 280 for more USS and Interpret table information.

In the following example:

- The system administrator from 1.1.1.1 will always be presented with a USSMSG10 screen from the USS table named USSTAB1. The entered command will be processed by the INTTAB1 table (if the command is in that table). Otherwise, the command will be processed by the USSTAB1 table.
- Clients identified in IPGROUP IPGPAY will be put in session with the PAYROLL application (if connected in TN3270 or TN3270E mode).
- Clients identified in HNGROUP HNGINV will be put in session with the INVENTRY application regardless of connection mode. The DEFONLY parameter blocks the end user from specifying any other application on an error screen.
- Clients connecting to LINK1 as the destination IP address and not matching IP address 1.1.1.1, IPGROUP IPGPAY, or HNGROUP HNGINV will be assigned an LU from the DEFAULTLUS pool and will be put in session with the logon manager application, TPX1. The same Client Identifiers on DEFAULTAPPL and USSTCP statements indicate that both a default application and a MSG10 screen are desired. Telnet cannot provide both, and it gives DEFAULTAPPL a higher priority. After the initial session attempt, the assigned USS table is used for USS messages if statement MSG07 is coded. For example, if an error occurs during logon to TPX1, a USSMSG from USSTAB1 will be sent to the client if MSG07 is coded. Otherwise, the connection is dropped. See “Connection and Session Persistence” on page 283 for more information. A client from 1.1.1.1 could connect to destination LINK1. However, an exact IP address on a mapping statement has higher priority than a linkname on a mapping statement. Therefore, the client would receive a USSMSG10 screen instead of being in session with TPX1. See “Client Identifier Selection Rules” on page 249 for exact rules.

```
IPGROUP IPGPAY
   1.1.2.2  1.1.2.3
   255.255.0.0:2.2.0.0
ENDIPGROUP
USSTCP  USSTAB1 1.1.1.1
INTERPTCP INTTAB1 1.1.1.1
DEFAULTAPPL PAYROLL  IPGPAY
LINEMODEAPPL INVENTRY HNGINV DEFONLY
DEFAULTAPPL INVENTRY HNGINV DEFONLY
DEFAULTAPPL TPX1  LINK1
USSTCP USSTAB1 LINK
```

Generic and Specific Device (LU Name) Pools: The TN3270 Telnet server supports Generic and Specific connection requests over TN3270E connections. All TN3270 connections are considered Generic requests. The following Telnet statements are used:

- LUMAP parameters GENERIC/SPECIFIC
- DEFAULTLUSSPEC

A Generic TN3270E connection request from a client is similar to a TN3270 connection request. In both cases, the Telnet server is responsible for assigning an LU name to represent the client. However, on TN3270E connections, the server must give the assigned LU name to the client during device type negotiation. For TN3270 connections, the LU name is not assigned until negotiations are complete and an application has been chosen. See “LU Mapping Selection Rules” on page 250 for exact rules.

A Specific TN3270E connection request from a client includes the LU name the client wants the server to assign. The server confirms the choice during device type negotiation. If the server assignment algorithms reject the client choice, the server sends a device type reject to the client. Most clients then notify the end user that the requested LU name is not valid.

Requesting a specific LU name allows a client to be assigned the same LU every time. This is important if the host application is LU name dependent, and the client does not have a constant IP address or host name. To prevent the server from assigning these LU names to Generic clients, pools of LUs are designated as Generic or Specific. For a Specific request, Telnet first searches any Specific pools mapped to the Client Identifier and then searches any Generic pools mapped to the Client Identifier. If no pools are mapped, the default Specific LU pool is searched. For a Generic request, Telnet searches any Generic pools mapped to the Client Identifier. If no pools are mapped, the default Generic LU pool is searched. This search order safeguards LUs in the Specific LU pool from being assigned to Generic requests.

The LUMAP statement is used to define an LU name or pool (LUGROUP) name as Generic or Specific. The DEFAULTLUSSPEC statement is used to define the default Specific request pool. In the example below, any client from the Client Identifier IPGPAY must specify an LU name to be assigned an LU from LUGROUP LUGPAY. If that same client did not specify an LU name, Telnet assigns an LU from LUGINV1. This assignment method might be preferred if the payroll application function is LU name dependent, but the inventory application provides the same function regardless of which LU name is assigned. If a client connects specifying an LU name and is not mapped to any LU pool, the LU name must be within the DEFAULTLUSSPEC LU pool.

```
IPGROUP IPGPAY
   1.1.2.2  1.1.2.3
   255.255.0.0:2.2.0.0
ENDIPGROUP
HNGROUP HNGINV
   INV1.DEPT1.GROUP1.COM
   *.DEPT3.GROUP1.COM
   **.GROUP3.COM
ENDHNGROUP
DEFAULTLUSSPEC
   LUDFS01  LUDFS02  LUDFS03
   LUDFS10..LUDFS99
ENDDEFAULTLUSSPEC
LUGROUP LUGPAY
   LUPAY01,LUPAY02,LUPAY03,LUPAY04,LUPAY05
```

```
   LUPAY06.........LUPAY20
ENDLUGROUP
LUMAP LUGPAY 1.1.2.1
LUGROUP LUGINV1
   LUINV01..LUINV20
ENDLUGROUP
LUMAP  LUGPAY    IPGPAY SPECIFIC
LUMAP  LUGINV1   IPGPAY GENERIC
LUMAP  LUGINV1   HNGINV GENERIC
```

Because the LU pool (LUGROUP) is defined on the LUMAP statement, a single LU pool can be both Specific and Generic, depending on the Client Identifier. For example, if the LUGINV1/HNGINV map were Specific instead of Generic, the LUGINV1 pool would be Generic for IPGPAY clients and would be Specific for HNGINV clients. Conflicts can occur if Telnet assigns an LU to the Generic IPGPAY client that is specifically wanted by an HNGINV client. Even though HNGINV specified the LU name it is already in use by the IPGPAY client on a Generic TN3270E connection and will result in a failed specific connect request.

Another feature of Specific LU name requests is that the client can specify an LUGROUP name, and the server will assign an available LU from that pool. This capability is useful when different applications require different LU naming schemes, but each end user client does not need to use the exact same LU name for each connection. For example, an administrator can create three pools, one for each of three applications. Only three client emulators need to be set up. One for TSO which requests LU name LUTSO, one for CICS which requests LUCICS, and one for IMS which requests LUIMS. Assume the general users are in subnet 3.0.0.0. Any client connecting with a Client Identifier of IPGGEN can specify LUTSO, LUCICS, or LUIMS and will be assigned an LU from the appropriate pool.

```
IPGROUP  IPGGEN
     255.0.0.0:3.0.0.0
ENDIPGROUP
LUGROUP  LUTSO
     TSO00001..TSO00999
ENDLUGROUP
LUGROUP  LUCICS
     CICS0001..CICS0999
ENDLUGROUP
 LUGROUP  LUIMS
     IMS00001..IMS00999
ENDLUGROUP
LUMAP LUTSO   IPGGEN  SPECIFIC
LUMAP LUCICS  IPGGEN  SPECIFIC
LUMAP LUIMS   IPGGEN  SPECIFIC
```

Pool name specification is a powerful mapping method because multiple LUMAP statements with different Objects can be used for a single Client Identifier. See “Additional LUMAP Functions” on page 244 for more information about the benefits of multiple LUMAP statements.

Printers: The Telnet server supports client printer emulation over TN3270E connections. Mapping statements for printer LUs are functionally identical to mapping statements for terminal LUs. Replace ’PRT’ with ’LU’ in the following statements to equate them to the terminal LU statements discussed earlier. The server supports Generic, Specific, and Associated connection requests with the following mapping statements:

- PRTMAP
- PRTGROUP
- DEFAULTPRT

- DEFAULTPRTSPEC
- LUMAP for association

Generic and Specific printer LU connection requests are supported by the first four statements above in exactly the same manner terminal LUs are supported. In the example below, a Generic default printer LU pool is defined to support printer clients that can use any name. A Specific default printer LU pool is defined to support mobile printer clients that must always use the same name. Another set of printers must be dedicated to inventory employees. The PRTMAP statement assigns the LUs in PRTGROUP PRTGINV to any client identified by HNGINV. PRTGINV is defined as a Generic pool on the PRTMAP statement. Using the Specific/Generic search rules an inventory end user could:

- Specify an exact printer LU name from the PRTGINV pool and be assigned that LU
- Specify the pool name and be assigned any LU in the pool
- Specify no name and be assigned any LU in the pool

In all three cases, the chosen LU would be from the PRTGINV pool.

```
DEFAULTPRT
    PRTDF01  PRTDF02  PRTDF03
    PRTDF10..PRTDF19
 ENDDEFAULTPRT
 DEFAULTPRTSPEC
     PRTDFS01  PRTDFS02  PRTDFS03
     PRTDFS10..PRTDFS99
 ENDDEFAULTPRTSPEC
 HNGROUP  HNGINV
     INV1.DEPT1.GROUP1.COM
     *.DEPT3.GROUP1.COM
     **.GROUP3.COM
 ENDHNGROUP
 PRTGROUP  PRTGINV
     PRTINV01  PRTINV02  PRTINV03  PRTINV04  PRTINV05
     PRTINV06..PRTINV20
 ENDPRTGROUP
 PRTMAP  PRTGINV  HNGINV
```

The associated printer LU connection request is unique to printers. It allows a printer to specify an active LU terminal name during connection negotiation. The server understands this special request and knows to assign a printer LU name that is associated with the requested terminal LU name. The association is established by linking a terminal LU pool (LUGROUP) with a printer LU pool (PRTGROUP). The two LU pools MUST have the same number of LUs defined so the LUs can be paired. If the pools do not have the same number of LUs defined, error messages will be produced during profile processing and during associated connect requests. Association can only occur when one-to-one pairings are done. For example, once the pools are linked Telnet will assign the third printer LU to a printer connection that requests association with the third terminal LU. Telnet will assign the 32nd printer LU to a printer connection that requests association with the 32nd terminal LU. The pools are linked with the LUMAP statement. The printer LU pool name is linked to the terminal LU pool name by adding the PRTGROUP name after the GENERIC/SPECIFIC option on the LUMAP statement. For example, the payroll application has print capability and will automatically send print data to a certain printer based on the terminal LU processing the request. In this case, it is very important that the same printer LU always be associated with the same terminal LU. Whenever LUPAY04 is connected, an associated printer request from that emulator

program will cause Telnet to assign PRTPAY04 to the printer client. When a print request is processed, the data will go to the correct printer. If the requested device is already in use, the request is rejected.

```
PRTGROUP  PRTGPAY
     PRTPAY01  PRTPAY02  PRTPAY03  PRTPAY04  PRTPAY05
     PRTPAY06..PRTPAY20
 ENDPRTGROUP
 LUMAP  LUGPAY  IPGPAY SPECIFIC  PRTGPAY
```

Additional LUMAP Functions: The DEFAPPL parameter on the LUMAP statement allows a host application to be mapped with an LU name or LUGROUP name instead of using DEFAULTAPPL. The LUMAP-DEFAPPL statement is treated just like DEFAULTAPPL when a Client Identifier matches the LUMAP statement. The LUMAP-DEFAPPL statement also supports the LOGAPPL, FIRSTONLY, and DEFONLY parameters that are used by DEFAULTAPPL and LINEMODEAPPL. See "Application Security" on page 253 for more details about these parameters. The LUMAP-DEFAPPL statement is a powerful statement when combined with multiple LUMAP statements for the same Client Identifier. Some examples are:

- Payroll clients already specify an LU name that can only be used with the PAYROLL application. Adding DEFAPPL is like adding a DEFAULTAPPL statement. The LOGAPPL parameter gives the added advantage of handling an inactive application. VTAM will continue session initiation after the application is active.
- General use clients have been set up to issue a Specific request for LU pool LUTSO, LUCICS, or LUIMS. After an LU is assigned, the DEFAPPL parameter will cause Telnet to immediately issue a session request for the appropriate application. Again, if LOGAPPL is coded and the application is not active, VTAM will continue session initiation later.
- The FIRSTONLY parameter affects what happens after LOGOFF. LOGOFF of a CICS or IMS session will result in a redrive to the application if LUSESSIONPEND is coded. LOGOFF of a TSO session will result in a USSMSG10 screen or solicitor panel begin sent to the client.
- In most cases DEFAPPL on Generic multiple LUMAPs is not useful. LUs are assigned in order starting with the first LUMAP statement. One case that may be useful is if an application has a user limit but can be cloned. Assume the INVENTRY application can support only 20 users but can be cloned. Multiple LUMAPs with DEFAPPL will direct the first 20 HNGINV clients to INVENTRY, the next 20 HNGINV clients to INVENTR2, and the next 20 HNGINV clients to INVENTR3.
- DEFONLY is used to prevent the end user from logging on to any other applications.

```
LUGROUP   LUGINV1
  LUINV01..LUINV20
ENDLUGROUP
LUGROUP   LUGINV2
  LUINV21..LUINV40
ENDLUGROUP
LUGROUP   LUGINV3
  LUINV41..LUINV60
ENDLUGROUP
LUMAP   LUGPAY    IPGPAY   SPECIFIC  DEFAPPL  PAYROLL                        PRTGPAY
LUMAP   LUTSO     IPGGEN   SPECIFIC  DEFAPPL  TSO      LOGAPPL  FIRSTONLY
LUMAP   LUCICS    IPGGEN   SPECIFIC  DEFAPPL  CICS     LOGAPPL
LUMAP   LUIMS     IPGGEN   SPECIFIC  DEFAPPL  IMS      LOGAPPL
LUMAP   LUGINV1   HNGINV            DEFAPPL  INVENTRY LOGAPPL  DEFONLY
LUMAP   LUGINV2   HNGINV            DEFAPPL  INVENTR2 LOGAPPL  DEFONLY
LUMAP   LUGINV3   HNGINV            DEFAPPL  INVENTR3 LOGAPPL  DEFONLY
```

Some host applications are set up to inquire if a secondary LU is active. If the LU is active the application will initiate a session. In order for this to work the secondary LU must be active. The LUMAP parameter, KEEPOPEN, will cause the ACB to remain open for any LU assigned to the client by this mapping statement. KEEPOPEN and LOGAPPL are mutually exclusive. See “Timers” on page 290 for information about ending idle KEEPOPEN connections and “Connection and Session Persistence” on page 283 for KEEPOPEN details.

Mapping Additional Parameters: Connection parameters are typically defined once at the port level. Sometimes it is useful to have different connection parameters depending on the Client Identifier. The PARMSGROUP and PARMSMAP statements allow connection parameters to be defined at the Client Identifier level. This level of granularity applies to the following parameters:

- DEBUG
- CLIENTAUTH
- ENCRYPTION
- CONNTYPE

Assume the PAYROLL department is assigned the highest level of security and connections are being monitored with summary debug messages, general users are assigned negotiable security, and inventory employees are experiencing intermittent problems with Telnet connections requiring detailed debug messages. The following statements assign the security and debug levels to the areas needed and do not affect other areas. See “Connection Security” on page 255 for security information and “Telnet Diagnostics” on page 291 for debug information.

```
PARMSGROUP   PRMGDBG
  DEBUG DETAIL
ENDPARMSGROUP
PARMSGROUP   PRMGSEC1
  CONNTYPE SECURE
  ENCRYPTION SSL_3DES_SHA  ENDENCRYPTION
  DEBUG SUMMARY
ENDPARMSGROUP
PARMSGROUP   PRMGSEC2
  CONNTYPE NEGT
  ENCRYPTION SSL_RC4_MD5   ENDENCRYPTION
ENDPARMSGROUP
PARMSMAP   PRMGDBG  HNINV
PARMSMAP   PRMGSEC1 IPGPAY
PARMSMAP   PRMGSEC2 IPGGEN
```

LU Assignments Based on Application Name: In some cases, only certain LU names are eligible to be in session with the host application. Or only certain LU names are eligible to represent user IDs. The LU and LUG parameters on the ALLOWAPPL and RESTRICTAPPL statements provide this checking function and allow some LU name mapping based on application name. For example, assume the only LUs eligible to use the inventory set of applications are the LUs in the inventory LU pools. A new LUGROUP pool named LUGINVT contains LUs from LUGINV1, LUGINV2, and LUGINV3. The ALLOWAPPL statement requires that any session request to the inventory applications have an LU name defined in LUGINVT. The LUG parameter must be used carefully. When specified, Telnet must match the LU using both the common mapping algorithms and the mapping by application. For RESTRICTAPPL, assume security authorization is required to get to the PAYROLL application and only the PAYxx user IDs and the system administrator are authorized and those user IDs map to certain LUs.

```
LUGROUP   LUGINV1
  LUINV01..LUINV20
ENDLUGROUP
LUGROUP   LUGINV2
  LUINV21..LUINV40
ENDLUGROUP
LUGROUP   LUGINV3
  LUINV41..LUINV60
ENDLUGROUP
LUGROUP   LUGINVT
  LUINV01..LUINV60
ENDLUGROUP
ALLOWAPPL   INVENTR*  LUG   LUGINVT
RESTRICTAPPL   PAYROLL
  USER  SYSADMIN  LU    LUADMNM
  USER  PAY01     LU    LUPAY01
  USER  PAY02     LU    LUPAY02
        (user pay03 through pay20 not listed)
```

TN3270 connections do not assign an LU to represent the client until an application name is chosen. Therefore, the LU and LUG parameters can be used as sole LU mapping statements for TN3270 connections. For example, assume no other mapping statements exist (LUMAP or DEFAULTLUS) and no TN3270E connections will be used. The ALLOWAPPL statements below will map LUs to the appropriate application based on the application name chosen. The RESTRICTAPPL statement below will assign a single LU or LU pool to each user.

```
ALLOWAPPL   TSO*  LUG   LUGTSO
ALLOWAPPL   CICS  LUG   LUGCICS
ALLOWAPPL   IMS   LUG   LUGIMS
RESTRICTAPPL   APP*
  USER   USER01   LUG  LUG01
  USER   USER02   LU   LU02
  USER   USER03   LU   LU03
```

Both of these assignment methods were very popular before TN3270E connections were introduced. These statements are not included in the combined profile below because TN3270E connections will likely achieve poor mapping results. An LU must be assigned during connection negotiation before the application name is known which will likely result in an LU mismatch later. TN3270E connections require that either a default LU pool or LUMAP statement exists because an LU must be assigned to the connection during negotiations before an application name is known. Consider the following example:

```
DEFAULTLUS
  LU1 LU2 LU3 LU4
ENDDEFALTLUS
RESTRICTAPPL  APPL1
  USER  USER3  LU  LU3
ALLOWAPPL   APPL2  LU  LU4
```

Assume two TN3270 connections are started.

- Two solicitor screens appear.
- Specify APPL1, USER3, and a password. The server selects LU3 based on both the DEFAULTLUS and the RESTRICTAPPL statements.
- Specify APPL2. The server selects LU4. The server selects LU4 based on both the DEFAULTLUS and the ALLOWAPPL statements.

Assume two TN3270E connections are started.

- Two solicitor screens appear. The server assigns LU1 and LU2.

- Specify APPL1, USER3, and a password. The server fails the connection because of an LU mismatch.
- Specify APPL2. The server fails the connection because of an LU mismatch.

If LU name mapping by application name or user ID is desired with TN3270E clients, the following three solutions are available:

- If the same application or user ID is always used at the same client, individual LUMAP statements can be used to map the correct LU name to each client. Then every connection request will result in the correct LU assignment for that client. The assumptions are that the client keeps the same IP address (or host name) and only one client exists at that IP address.
- The TELNETPARMS statement NOTN3270E disables all TN3270E function in the server so all connections will be TN3270, not TN3270E. The drawback is that all TN3270E function is disabled for the entire server. This includes printer function, Generic/Specific function, and SNA function to the client.
- The TELNETPARMS statement SIMCLIENTLU is a less severe solution. This function will send a dummy LU name of EZBSIMLU to all TN3270E clients issuing Generic connection requests to satisfy the negotiation but will not assign a Telnet LU until an application name is chosen. This alternative preserves printer function, Specific requests, and SNA function to the client. The drawback is the name sent to the client is not the name Telnet ultimately uses to represent the client. Printer association will not work for these TN3270E Generic connections and any emulator programming that depends on the LU name will be using the dummy LU name.

Combining the Assignment Statement Examples

The examples above can be combined to create a single assignment profile. Statements other than LU assignment statements are not shown here. See “Device Types and Logmode Considerations” on page 252 for TELNETDEVICE details. See “Connection and Session Persistence” on page 283 for MSG07 and LUSESSIONPEND details.

```
BEGINVTAM

DEFAULTLUS                    ; Terminal Generic Default pool
  LUDF01  LUDF02  LUDF03
  LUDF04..LUDF19
  LUDF100..LUDF999
ENDDEFAULTLUS
DEFAULTLUSSPEC                ; Terminal Specific Default pool
  LUDFS01  LUDFS02  LUDFS03
  LUDFS10..LUDFS99
ENDDEFAULTLUSSPEC
DEFAULTPRT                    ; Printer Generic Default pool
  PRTDF01  PRTDF02  PRTDF03
  PRTDF10..PRTDF19
ENDDEFAULTPRT
DEFAULTPRTSPEC                ; Printer Specific Default pool
  PRTDFS01  PRTDFS02  PRTDFS03
  PRTDFS10..PRTDFS99
ENDDEFAULTPRTSPEC

LUGROUP   LUGPAY         ; Payroll LUs
  LUPAY01  LUPAY02  LUPAY03
  LUPAY04  LUPAY05
  LUPAY06..LUPAY20
ENDLUGROUP
LUGROUP   LUGINV1        ; Inventory LUs - pool 1
  LUINV01..LUINV20
ENDLUGROUP
LUGROUP   LUGINV2        ; Inventory LUs - pool 2
```

```
  LUINV21..LUINV40
ENDLUGROUP
LUGROUP   LUGINV3        ; Inventory LUs - pool 3
  LUINV41..LUINV60
ENDLUGROUP
LUGROUP   LUGINVT        ; Inventory LUs - Total
  LUINV01..LUINV60
ENDLUGROUP
LUGROUP   LUTSO          ; TSO LUs
  TSO00001..TSO00999
ENDLUGROUP
LUGROUP   LUCICS         ; CICS LUs
  CICS0001..CICS0999
ENDLUGROUP
LUGROUP   LUIMS          ; IMS LUs
  IMS00001..IMS00999
ENDLUGROUP
PRTGROUP   PRTGINV       ; Printer LUs for Inventory
  PRTINV01  PRTINV02  PRTINV03
  PRTINV04  PRTINV05
  PRTINV06..PRTINV20
ENDPRTGROUP
PRTGROUP   PRTGPAY       ; Printer LUs for Payroll
  PRTPAY01  PRTPAY02  PRTPAY03
  PRTPAY04  PRTPAY05
  PRTPAY06..PRTPAY20
ENDPRTGROUP

IPGROUP  IPGPAY         ; IP addresses for Payroll
     1.1.2.2  1.1.2.3
     255.255.0.0:2.2.0.0
ENDIPGROUP
IPGROUP  IPGGEN          ; General Connections
   255.0.0.0:3.0.0.0
ENDIPGROUP
HNGROUP HNGINV
   INV1.DEPT1.GROUP1.COM
   *.DEPT3.GROUP1.COM
   **.GROUP3.COM
ENDHNGROUP

PARMSGROUP  PRMGDBG
     DEBUG DETAIL
ENDPARMSGROUP
PARMSGROUP  PRMGSEC1
   CONNTYPE SECURE
   ENCRYPTION SSL_3DES_SHA  ENDENCRYPTION
   DEBUG SUMMARY
ENDPARMSGROUP
PARMSGROUP  PRMGSEC2
   CONNTYPE NEGT
   ENCRYPTION SSL_RC4_MD5   ENDENCRYPTION
ENDPARMSGROUP
PARMSMAP  PRMGDBG  HNINV
PARMSMAP  PRMGSEC1 IPGPAY
PARMSMAP  PRMGSEC2 IPGGEN

LUMAP LUADM 1.1.1.1
LUMAP LUADMNM ADMIN.DEPT1.GROUP1.COM
LUMAP LUGPAY 1.1.2.1.

LUMAP   LUGPAY    IPGPAY   SPECIFIC  DEFAPPL  PAYROLL                        PRTGPAY
LUMAP   LUTSO     IPGGEN   SPECIFIC  DEFAPPL  TSO      LOGAPPL  FIRSTONLY
LUMAP   LUCICS    IPGGEN   SPECIFIC  DEFAPPL  CICS     LOGAPPL
LUMAP   LUIMS     IPGGEN   SPECIFIC  DEFAPPL  IMS      LOGAPPL
LUMAP   LUGINV1   HNGINV             DEFAPPL  INVENTRY LOGAPPL  DEFONLY
```

```
LUMAP   LUGINV2   HNGINV             DEFAPPL  INVENTR2 LOGAPPL  DEFONLY
LUMAP   LUGINV3   HNGINV             DEFAPPL  INVENTR3 LOGAPPL  DEFONLY
PRTMAP  PRTGINV  HNGINV
DEFAULTAPPL INVENTRY  HNGINV
DEFAULTAPPL TPX1      LINK1
DEFAULTAPPL PAYROLL   IPGPAY
LINEMODEAPPL INVENTRY HNGINV

USSTCP USSTAB1 LINK1
USSTCP USSTAB1 1.1.1.1
INTERPTCP INTTAB1 1.1.1.1

RESTRICTAPPL  PAYROLL
   USER  SYSADMIN LU  LUADMNM
   USER  PAY01    LU  LUPAY01
   USER  PAY02    LU  LUPAY02
          (user pay03 through pay20 not listed)

ALLOWAPPL  INVENTR* LUG  LUGINVT
ALLOWAPPL *

ENDVTAM
```

Client Identifier Selection Rules

When Client Identifiers are used together, conflicts might occur. For example, host name NAME1.HOST1.COM may be IP address 1.2.3.4. If the following DEFAULTAPPL statements exist, only one application can be chosen.

```
DEFAULTAPPL  TSO  NAME1.HOST1.COM
DEFAULTAPPL  CICS 1.2.3.4
```

Telnet has very specific rules to identify the hierarchy of Client Identifiers for each mapping statement. The following order is used for each mapping statement:

1. Exact host name on any mapping statement. For example, a client with host name NAME1.HOST1.COM connects.

   ```
   LUMAP   LU1         NAME1.HOST1.COM
   ```

2. Exact IP address on any mapping statement. For example, a client with IP address 1.2.3.4 connects.

   ```
   LUMAP    LU2       1.2.3.4
   ```

3. Exact host name in an HNGROUP on any mapping statement. For example, a client with host name NAME2.HOST1.COM connects.

   ```
   HNGROUP  HNGRP1
        NAME2.HOST1.COM  NAME2.HOST3.COM
        *.HOST2.COM
        **.HOST3.COM
   ENDHNGROUP
   LUMAP   LUGRP1   HNGRP1
   ```

4. Exact IP address in an IPGROUP on any mapping statement. For example, a client with IP address 1.2.3.5 connects.

   ```
   IPGROUP  IPGRP1
     1.2.3.5     1.2.3.6
     255.255.0.0:2.3.0.0
   ENDIPGROUP
   LUMAP   LUGRP1   IPGRP1
   ```

5. Most specific wildcard host name match in an HNGROUP on any mapping statement. For example, from the group defined above, a client with host name NAME1.HOST2.COM connects.
6. Most specific IP subnet:mask match in an IPGROUP on any mapping statement. For example, from the group defined above, a client with IP address 2.3.1.1 connects.

7. Linkname match on the following mapping statements.

```
DEFAULTAPPL    TSO        LINK1
LINEMODEAPPL   CICS       LINK1
USSTCP         USSTAB1    LINK2
INTERPTCP      INTTAB1    LINK2
PARMSMAP       PRMSGRP1   LINK2
```

8. ″Null″ name on the following mapping statements.

```
DEFAULTAPPL     TSO
LINEMODEAPPL    CICS
USSTCP          USSTAB1
INTERPTCP       INTTAB1
```

LU Mapping Selection Rules

LU mapping selection can become complicated because of the many variations of mapping statements, TN3270E versus TN3270 connections, Generic versus Specific connection requests, printer association, and LU mappings based on application name. LU mapping is very different between TN3270E and TN3270 and will be discussed separately.

General Mapping Rules for both TN3270E and TN3270 follow:

- If multiple LUMAP statements exist for a Client Identifier all Specific LUMAPs are searched (TN3270E only) and then all Generic LUMAPs are searched in the order they are listed in the profile.
- If the application is known during the LU lookup and the ALLOWAPPL or RESTRICTAPPL-USER statement has LUs listed, then the found LU must be in both the LUMAP or default LU group and in the application LU group.
- After an LU match is found, the search stops.
- Telnet performs database lookup for Objects based on the Client Identifier. TN3270E connections require an Early Lookup so Telnet can give the client an LU name during connection negotiation. In all cases a Complete Lookup is done when the application name is given. Telnet performs an Early Lookup and a Complete Lookup for TN3270E connections. Telnet performs only a Complete Lookup for TN3270 and Linemode connections.

TN3270E LU Mapping: TN3270E connections require an Early Lookup during connection negotiation. Telnet will use as much information as is available to map an LU to the client. However, the eventual application is not known at this time unless an LUMAP-DEFAPPL or DEFAULTAPPL statement defines the application name. After connection negotiations are complete, Telnet will either send a logon solicitor (or USSMSG10) screen to the client or will perform a Complete Lookup using the application name obtained through the LUMAP-DEFAPPL or DEFAULTAPPL statement. If Complete Lookup is successful, Telnet will begin session establishment. If a solicitor (or USSMSG10) screen is sent to the client, an application name must be entered, at which time Telnet will perform a Complete Lookup. If LU mapping is being done based on application name, a conflict might occur between the application LU mapping and the LU already assigned to the connection. For TN3270E, once an LU name is assigned during connection negotiation it can never change until the connection is dropped. The SIMCLIENTLU statement allows Telnet to map LUs for TN3270E connections as though they were TN3270 connections. See “TN3270 LU Mapping” on page 251 for mapping Generic TN3270E connection requests with SIMCLIENTLU. A request for a specific LU from the Telnet Client will be treated as if SIMCLIENTLU was not specified. The exact lookup process for TN3270E (non- SIMCLIENTLU) is described below.

Early Lookup - An LU must be found during Early Lookup. LUMAP-DEFAPPL and DEFAULTAPPL statements are considered but not necessarily used. Possible lookup results are:

- An LU is found.
- An LU is not found, the connection is dropped.

Perform TN3270E Early Lookup in the following order.

- Check for LUMAP matches considering application lookup results and possible application-based LU mappings.
  1. For each Specific LUMAP - If the Specific LUMAP has DEFAPPL, or DEFAULTAPPL was specified and the application lookup return code is either OK or USER_REQUIRED then perform LU lookup.
  2. For each Generic LUMAP - If the Generic LUMAP has DEFAPPL, or DEFAULTAPPL was specified and the application lookup return code is either OK or USER_REQUIRED then perform LU lookup.
- Check for LUMAP matches without considering application lookup results.
  1. For each specific LUMAP - Perform LU lookup ignoring DEFAPPL and DEFAULTAPPL.
  2. For each generic LUMAP - Perform LU lookup ignoring DEFAPPL and DEFAULTAPPL.
- If LUMAP statements were ″not checked″ (different from ″checked but no match″), use the appropriate Default LU pool considering application lookup results and possible application-based LU mappings. In this case the only relevant application is the defaultappl. If the application lookup return code is either OK or USER_REQUIRED then perform LU lookup.
- If no LUMAP statements were checked, try the appropriate Default LU pool without considering application lookup results. Perform LU lookup.

Complete Lookup - An application name is required for Complete Lookup. The application name is obtained from one of three sources in the order specified.

1. Input from the USER or VTAM (via CLSDST with OPTCD=PASS).
2. DEFAPPL parameter on the LUMAP statement.
3. DEFAULTAPPL statement.

Use the application name and the previously found LU to perform Complete Lookup. Possible lookup results are:

- The application is not valid.
- The application is valid (return code OK or USER_REQ) for the existing LU.
- The application-based LU map does not match the already chosen LU.

TN3270 LU Mapping: TN3270 connections only perform Complete Lookup after all information is known. LU lookup is not done during connection negotiation. Telnet will either send a solicitor (or USSMSG10) screen to the client or will perform Complete Lookup using the application name known through the LUMAP-DEFAPPL or DEFAULTAPPL statement. If Complete Lookup is successful, Telnet will begin session establishment. If not successful, the solicitor (or USSMSG10) screen is sent to the client without an LU being assigned to the connection. The LU is not assigned until the application name is known. If the application name is a RESTRICTAPPL, the LU is not assigned until a user ID is specified. Application-based LU mappings have a very good chance of success due to the late LU mapping aspect of TN3270 connections. When SIMCLIENTLU is coded, Generic TN3270E connections have this same characteristic.

Complete Lookup - An application name is required for Complete Lookup. The application name is obtained from one of three sources in the order specified.

1. Input from the USER or VTAM (CLSDST with OPTCD=PASS).
2. DEFAPPL parameter on the LUMAP statement.
3. DEFAULTAPPL statement.

Use the application name to perform Complete Lookup. Possible lookup results are:

- The application is not valid.
- The application is valid but an LU is not found.
- The application is valid (return code OK) and an LU is found.
- The application-based LU map does not match the Client Identifier LU.

If the application is not valid, no LU is assigned to the connection and an error message is sent to the client. If the application is valid continue the LU lookup in the following order.

- Check for LUMAP matches considering application-based LU lookup results. Only Generic LUMAPs are searched. If the Generic LUMAP has LOGAPPL, or DEFAULTAPPL was specified and the application lookup return code is OK then perform LU lookup.
- Check for application-based LU mappings. If the application lookup return code is OK and LUs are defined on the application statement perform LU lookup.
- If no LUMAP statements were checked, use the appropriate Default LU pool considering application lookup results. If the application lookup return code is OK, then perform LU lookup.

Device Types and Logmode Considerations

The VTAM logmode defines many characteristics of the session established between the Telnet LU representing the client and the host application. For example, the logmode defines response types, presentation style, and the type of LU Telnet is emulating. LU0 (non-SNA) and LU2 (SNA) represent terminal LU types. LU1 (SCS) and LU3 (3270 Data) represent printer LU types.

Telnet matches a VTAM logmode to each client as it connects based on the client device type. Refer to *z/OS Communications Server: IP Configuration Reference* for Device Type and Logmode Table information. The default terminal logmodes are non-SNA for TN3270 connections and SNA for TN3270E connections. At session request time, Telnet indicates to VTAM the desired logmode based on device type. The host application usually honors the request and binds the session using the requested logmode. However, depending on VTAM statements, the host application can override the requested logmode and bind the session using different characteristics than Telnet requested. For this reason, some screen sizes might not work correctly even though the logmode defined in Telnet is correct. If the KEEPOPEN function is used to allow session initiation by the host application, the desired logmode must be coded on the DLOGMOD parameter as part of the VTAM application definition statement that defines the Telnet LU. Otherwise, the host application will choose its own logmode.

Telnet processes the ATTN KEY request differently for non-SNA and SNA sessions. For non-SNA sessions (BIND FM value 02), Telnet converts the ATTN KEY request to a ’6C’x data byte and sends it to the application. For SNA sessions (BIND FM value 03), Telnet converts the ATTN KEY request into a SNA signal and sends it to the host application as expedited data. Some clients send both an ATTN KEY function code and a ’6C’x data byte to ensure the ATTN is seen by the application.

Telnet converts the ATTN KEY function into either a '6C'x data byte or a SNA signal and also forwards the '6C'x data. Some applications give unexpected results or Telnet might appear to not support ATTN when two ATTNs are received. The SINGLEATTN statement in TELNETPARMS causes Telnet to drop the second ATTN if it immediately follows an ATTN. SINGLEATTN is a one statement server solution to a problem found in potentially thousands of clients.

To change either the TN3270 or the TN3270E logmode for a device type, use the TELNETDEVICE statement in the BEGINVTAM block. Telnet has no input for printer session logmode choice because the host always initiates the session. In the examples that follow, the first line causes only the 3270 logmode to change from the default to SNX32705. The second line causes both the 3270 and 3270E logmodes to change from their defaults to SNX32705 and SNX32702. The third line causes only the 3270E logmode to change from the default to SNX32702.

```
TELNETDEVICE 3278-5-E SNX32705
TELNETDEVICE 3278-5-E SNX32705,SNX32702
TELNETDEVICE 3278-5-E SNX32702
```

Security

Telnet provides many levels of security. Telnet profile statements allow clients open access to certain applications, give restricted access by user ID to other applications, protect against data overruns, and provide several connection security options. Application security is administered by the ALLOWAPPL and RESTRICTAPPL statements in BEGINVTAM. Data overrun protection is administered by MAXRECEIVE, MAXVTAMSENDQ, and MAXREQSESS statements in TELNETPARMS. Connection security options are set using a variety of Telnet statements which all fall under the Secure Socket Layer (SSL) function.

Application Security

The ALLOWAPPL statement in BEGINVTAM indicates which host applications a client is ″allowed″ to access. Wildcard names are permitted and are very useful when defining applications that issue CLSDST PASS to another similarly named application. For example, a connection to TSO immediately results in a CLSDST PASS from TSO to TSO0001, or something similar. The wildcard statement below is necessary to allow clients to connect to TSO.

```
ALLOWAPPL   TSO*
```

The RESTRICTAPPL statement in BEGINVTAM indicates which host applications a client can access but ″restricts″ access to those with approved user IDs and passwords. Wildcard names are permitted on both the application and the user ID. A restricted application name can be chosen three different ways:

- The client is mapped to a DEFAULTAPPL or LUMAP-DEFAPPL
- The client enters the name on the Telnet Solicitor Panel
- The client enters the name on a USS message panel.

Whichever way the application name is chosen, Telnet discovers the name is a restricted application and sends a Solicitor Panel to the client requesting a user ID and password. The user ID for the application is verified by Telnet. Once the user ID is validated and a Password is obtained, Telnet submits the user ID/Password pair for authorization to a security program such as RACF. If authorized, Telnet continues with session setup. In the example below, the PAYROLL application is restricted to user SYSADMIN and users PAY01-PAY20. Any other user attempting to access the PAYROLL application will be rejected by Telnet.

```
RESTRICTAPPL   PAYROLL
  USER   SYSADMIN
  USER   PAY01
  USER   PAY02
    (user pay03 through pay20 not listed)
```

When searching for a match with the input application name, Telnet will find the most specific match whether it is on the ALLOWAPPL or RESTRICTAPPL statement. If each statement has the same name specified, the RESTRICTAPPL entry is used. For example, TSO has its own user ID/password requirement and probably does not need the additional Telnet security check. However, the Telnet security check may be needed for all other applications. This example can be supported with the following statements.

```
RESTRICTAPPL   *
  USER   *
ALLOWAPPL   TSO*
```

The LU or LUG parameter can be coded on the ALLOWAPPL statement and on each USER statement. Multiple LUs can be assigned individually using the LU keyword or a single LU group can be assigned using the LUG parameter. LU and LUG cannot be mixed on a single statement and only one LUG entry per statement is permitted. LU assignment based on application is a convenient way to limit the access to applications. However, this increases mapping complexity significantly when LU mapping statements and connection types are part of the overall mapping equation. See “LU Assignments Based on Application Name” on page 245 for more details about LU and LUG use.

DEFAULTAPPL, LINEMODEAPPL, and LUMAP-DEFAPPL statements in BEGINVTAM define a default application based on the Client Identifier. Instead of giving the end user an application selection screen, the profile can be coded to assign a default application based on the Client Identifier. When coded, Telnet will immediately initiate a session request when the connection is accepted. DEFAULTAPPL is used to define the default application for TN3270, TN3270E, and Linemode-Transform connections. LINEMODEAPPL is used to define the default application for Linemode-Standard and Linemode-Binary connections. LUMAP-DEFAPPL is used to define a default application for any connection type when an LUMAP statement matches the Client Identifier. LUMAP-DEFAPPL takes priority over DEFAULTAPPL or LINEMODEAPPL. See “Mapping Statements” on page 238 for usage examples.

Assigning a default application blocks the end user from entering their own application choice. The DEFONLY parameter and the MSG07 and LUSESSIONPEND statements also affect the level of security provided by the default statement.

DEFONLY, FIRSTONLY, and LOGAPPL are parameters on DEFAULTAPPL, LINEMODEAPPL, and LUMAP-DEFAPPL. See “Advanced Persistence Topics” on page 284 for details about the LOGAPPL and FIRSTONLY parameters.

- If the default application is not available and no other statements are coded, the connection is dropped. The end user cannot get to any other application other than the default. However, no error messages are sent to the end user and auto-reconnect loops are possible. For these reasons it is recommended that MSG07 always be used.
- If the LUMAP-DEFAPPL statement is coded and the default application is inactive, an error screen will be sent to the client whether or not MSG07 is coded.

- If MSG07 or LUMAP-DEFAPPL is coded and the default application is inactive, an error screen will be sent to the client. The DEFONLY parameter will block a user-entered application choice if it is different than the default. This parameter prevents application choice while giving the end user error information.
- If the end user logs off a session and LUSESSIONPEND is not coded, the connection is dropped.
- Code LUSESSIONPEND to redrive the initial database lookup after session logoff. Later results will be identical to the first lookup. If a default application for the client exists, Telnet will immediately initiate another session request. Otherwise, a USSMSG10 screen or solicitor panel will be sent to the end user.
- Sometimes a default application is used at initial connection, but after LOGOFF a USSMSG10 or solicitor panel is more appropriate than redriving the default. In this case, code the FIRSTONLY parameter. This indicates the default should be used on the first session only. After a session has been established, any subsequent lookups will ignore the default and send the USSMSG10 screen or solicitor panel.

If DEFAULTAPPL, LINEMODEAPPL, or LUMAP-DEFAPPL applications are mapped to the client and no ALLOWAPPL or RESTRICTAPPL is coded, the application is assumed to be an ALLOWAPPL type application.

Data Overrun Security

The MAXRECEIVE statement in TELNETPARMS limits the number of bytes received from a client without an End Of Record (EOR) being received. If the data received exceeds the limit, the connection is dropped. This parameter protects against a client stuck in a send-data loop. In general, large file transfers will not be affected because the sender typically divides the file into smaller records that are sent. The receiver rebuilds the file when all of the smaller records are received.

The MAXVTAMSENDQ statement in TELNETPARMS limits the number of data segments (RPLs) queued to be sent to VTAM. If the queue size exceeds the limit, the connection is dropped. This parameter protects against using up large amounts of storage to hold data destined for a host application that is not receiving data.

The MAXREQSESS statement in TELNETPARMS limits the number of session requests received by Telnet in a 10 second period. For this parameter, a BIND received by Telnet defines a session request. If the number of BINDs received in a 10 second period exceeds the limit, an error is reported. This parameter protects against session logon loops that are possibly created by an automatic CLSDST-PASS to an inactive session. This parameter cannot protect against logon loops caused by an inactive default application and a client using auto-reconnect. The best protection against the auto-reconnect loop is to code the MSG07 statement which keeps the client from being disconnected. However, other applications can be chosen from the error screen returned to the end user. See to the DEFONLY parameter in “Application Security” on page 253 for security information when using MSG07.

Connection Security

TN3270 Secure Socket Layer Overview: The TN3270 server provides the ability to protect Telnet connections with Secure Sockets Layer (SSL) protocol. The SSL protocol provides: server authentication, data integrity, and optionally, client authentication and data encryption. In this chapter, a port that is configured to use the SSL protocol is referred to as a *secure port* or SECUREPORT.

A connection that does not use the SSL protocol is referred to as basic connection.

References to RACF apply to any other SAF-compliant security products which contain the required support.

The SSL protocol begins with a handshake. During the handshake, the client authenticates the server, the server optionally authenticates the client, and the client and server agree on how to encrypt and decrypt information.

In an SSL-encrypted session, all data is encrypted using the SSL protocol before it is sent to the client. Data received from the client is decrypted before the data is sent to other processes, such as VTAM. The flows between Telnet and VTAM are unchanged.

Prior to release 10, the TN3270 server assumed that any connection on a secure port would immediately enter into an SSL handshake. This was also the action expected by the existing levels of HOD and PCOMM. Starting in release 10, the Internet Engineering Task Force (IETS) TLS-based Telnet Security Draft is also supported. This Draft allows a TN3270 negotiation to determine if the client wants/supports SSL prior to beginning the SSL handshake. The default action that the TN3270 server will take for a secure port is to first attempt an SSL handshake. If the client does not start the SSL handshake within the time specified by SSLTIMEOUT, an attempt will be made to negotiated SSL (as defined by the TLS-based Telnet Security Draft). If the client responds that SSL is desired, the SSL handshake is started ; if the client rejects SSL, the connection will be closed. This allows installation to support both types of SSL clients without knowing which protocol the client is using. The default action can be changed by specifying the CONNTYPE parameter described later in this chapter.

The encryption algorithm that is used for the connection depends on a combination of the encryption algorithms the server supports and the encryption algorithms the client requests. During the SSL handshake the client send a list of encryption algorithm it is willing to use. The server determines the best match between client's list and the encryption algorithms supported by the server for the connection. If the server does not support any of the encryption algorithms requested by the client, the connection will be closed. The TN3270 server uses the SSL support provided by the System Secure Sockets Layer (System SSL) element of z/OS. The encryption algorithms supported by the TN3270 server are therefore dependent on the level of System SSL installed on the customer's machine. The following encryption algorithms are supported by the base level of System SSL: NULL, RC2 export, RC4 export, DES. The System SSL Level 3 feature is required for Triple DES and RC4 non-export (128 bit) encryption algorithms. The encryption algorithms supported by TN3270 server can be limited to a subset of the algorithms provided by System SSL by specifying the ENCRYPT/ENDENCRYPT block.

The SECUREPORT statement must be specified in the port's TELNETPARMS block in order make the SSL function available for the port. Beginning in release 10, a port defined as a SECUREPORT can be used for both SSL secured connections and basic (non-SSL) connections. As always, a port defined with the PORT statement can only be used for basic connections.

SSL requires a server certificate as part of its server authentication process. The server certificate and the Certificate Authority certificates are stored in a keyring (also referred to as a key database). The server's keyring can be created using the GSKKYMAN utility provided by the System Secure Socket Layer (System SSL) element of z/OS or by using RACF's certificate management support. The keyring is

defined to the TN3270 server using the KEYRING statement in either the TELNETPARMS block or the TELNETGLOBALS block. All SECUREPORTs must use the same keyring.

Additional information about the concepts of cryptography and SSL can be found at the following Web sites:

- http://home.netscape.com/eng/ssl3/
- http://www.verisign.com/repository/crptintr.html

Hardware Encryption Support: Encryption is provided either by BSafe software shipped with System SSL or by hardware. There is no TCP profile definition that controls whether the cryptographic hardware will be used for TN3270 secure ports. When the first TN3270 secure port is brought online, System SSL checks if ICSF is installed and active and if the hardware is enabled and loaded with the necessary Master Keys. If the hardware is not available at that time, all subsequent encryption is performed using software. If hardware is valid and ICSF is active at that time, the public key functions required during the SSL handshake and requests for encryption using DES and Triple-DES algorithms will be sent to the hardware. Otherwise, all cryptographic functions will be performed by software. Encryption requests using RC2 or RC4 algorithms are always performed by software. Also note that if ICSF subsequently becomes unavailable, System SSL will assume the hardware encryption is still wanted and encryption processing using DES or Triple-DES algorithms will fail until access to the hardware is restored. Subsequent session handshakes will fail also. If all secure ports are stopped (V TCPIP,,T,STOP,PORT=S), the check for the cryptographic hardware presence and validity will subsequently be done again when the next TN3270 secure port is brought online.

If hardware encryption is to be used, be sure that the user ID associated with the TCP stack has read access to the RACF CSFSERV class resources. If ICSF is available but TCP has not been given access to these resources, the SSL initialization may fail with message: EZZ6030I TELNET SSL UNAVAILABLE, UNABLE TO INITIALIZE SSL INTERFACE, RSN=nn. The reason code nn is likely to be 4 (bad password) because System SSL will attempt to use the hardware encryption during processing of the keyring.

The following CSFSERV resources (service-names) are accessed by System SSL:

1. CSFCKI Clear Key Import
2. CSFCKM Clear Key Import Multiple
3. CSFDEC DES and TripleDES Decipher
4. CSFENC DES and TripleDES Encipher
5. CSFOWH MD5 and SHA1 Hashing
6. CSFRNG Random Number Generate
7. CSFPKB RSA Key Token Build
8. CSFPKX RSA Public Key Extrace
9. CSFPKE RSA Public Key Encipher
10. CSFPKD RSA Private Key Decipher
11. CSFPKI RSA Key Import
12. CSFDSG Digital Signature Generate
13. CSFDSV Digital Signature Verify

The RACF administrator should permit the user ID associated with TCP to these resources. For example:

```
PERMIT service-name CLASS(CSFSERV) ID(tcp-userid) ACCESS(READ)
```

The MAXLEN installation option for hardware cryptographic determines the maximum length that can be used to encrypt and decrypt data using ICSF/MVS. Set the MAXLEN ICSF/MVS installation option to 65527 or greater because this is the maximum TCP/IP packet size.

Refer to *z/OS ICSF Administrator's Guide* for additional information on controlling who can use cryptographic keys and services.

Server Authentication: When using SSL to secure communications, the SSL authentication mechanism known as server authentication is used.

With server authentication, the Telnet server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the Telnet server to the client application. The Telnet server supplies the client with the Telnet server's X.509 certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure communication channel is established between the Telnet server and the SSL enabled client.

For server authentication to work, the Telnet server must have a private key and associated server certificate in the server keyring file. If the GSKKYMAN utility was used to create the keyring, a password stash file is also required.

To conduct commercial business on the Internet, you might use a widely known certificate authority (CA), such as VeriSign, to get a high assurance server certificate. For a relatively small private network within your own enterprise or group, you can issue your own server certificates, called self-signed certificates, for your own use.

Client Authentication: Client authentication provides additional authentication and access control checking using client certificates at the TN3270 server. This support prevents a client from seeing and getting past the USSMSG without an installation approved certificate.

Three levels of client authentication are provided by the TN3270 server.

The first level of support is to support SSL Client Authentication as defined by the SSL protocol. The client passes an X.509 certificate to the S/390 TN3270 Server as part of the SSL Handshake. To pass authentication, the Certificate Authority (CA) that signed the client certificate must be considered trusted by the server. That is, the certificate for the CA must be in the keyring used by the TN3270 Server on the S/390 and designated as trusted. Note that the value of this option alone is based on which CAs are considered trusted. If the CA is a public CA and the certificate is in an easily obtained class, anyone can obtain such a certificate and therefore passing SSL Client Authentication does not provide much value-add unless coupled with the RACF support described below. If the CA is controlled by the enterprise, then the client that possesses such certificate is at least known to the organization. Therefore some level of access control is provided in this case.

In addition to the checking done with the first level of client authentication support, the second level of support requires that the client certificate is registered with RACF (or other SAF compliant security product) and mapped to a user ID. The client certificate received during the SSL handshake is used to query the security

product to verify that the certificate maps to a user ID known to the system prior to issuing the USSMSG screen. This solution provides additional access control at the TN3270 server and ensures that the end user cannot get past the TN3270 Server and attempt access to the SNA subsystem unless he is known to have a valid user ID on the TN3270 system.

This second level of client authentication requires the following setup:

- Ensure that RACDCERT is defined as an authorized TSO command in the IKJTSOxx member.
- Activate the RACF DIGTCERT class, if registering the full client certificate or activate the DIGTNMAP class if using Certificate Name filtering: SETROPTS CLASSACT(classname).
- Register the client certificate with RACF or setup a RACF Certificate Name Filter. There are various ways to do this. The section on using self-signed client certificates later in this chapter shows one method. For a complete description of RACF management of digital certificates and options available, see the *z/OS SecureWay Security Server RACF Security Administrator's Guide*. If using RACF's Certificate Name Filtering with MultiID filters, TN3270 client authentication processing only matches filters that specify generic (*) criteria.
- Refresh the applicable RACF class after any changes.

This second level of checking returns a user ID that has been associated with the client's certificate and ensures that any user that does not have a certificate defined to the security product cannot gain access to the Telnet server. However, it also means that any user that has a certificate defined to the security product can access any Telnet port. The optional capability to restrict access to the TN3270 server on a port basis is also desirable.

The third level of checking, provides the capability to restrict access on a port basis using the SERVAUTH RACF class. Under this class, the customer can specify the user IDs that are allowed to connect into a specific Telnet port by using a RACF profile name in the following format:

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

where nnnnn is the port number with leading zeros.

For example, the profile name for a TCP stack called TCPCS running on a system called MVSA for port 992 would be EZB.TN3270.MVSA .TCPCS.PORT00992. If all systems will use the same access list and RACF generic profile checking is active for the SERVAUTH class, the following profile name could be used:

```
EZB.TN3270.* .TCPCS.PORT00992
```

The profile name can contain wildcards to the extent that the security product allows.

Another example is EZB.TN3270.MVS.TCPCS.PORT00023, the security product profile name for port 23 running on the TCP stack called TCPCS on system MVS.

To protect all ports with a single profile, the following security product profile name could be used: EZB.TN3270.MVS.TCPCS.PORT* (if the installation's security product supports wildcards in profile names).

All security product rules (for example wildcards, PROTECTALL, and so on) apply.

The user ID associated with the client certificate can then be checked against the SERVAUTH class profile entry for the Telnet port. The use of this RACF class is optional. If the SERVAUTH RACF class is active and a RACF profile for the port is defined, this level of RACF authorization will be verified prior to issuing the USSMSG screen. If the SERVAUTH class is not active or there is no RACF profile protecting the port, this indicates to the TN3270 server that this level of check is not required and the client is allowed to connect to Telnet as long as the client certificate was validated (as described above).

To restrict access on a port basis, the following RACF setup is needed and must be done by a user ID that has authority to issue the specified RACF commands:

- Activate the RACF SERVAUTH class, if not active:

  ```
  SETROPTS CLASSACT(SERVAUTH)
  ```

- Define the profile for the port:

  ```
  RDEFINE  SERVAUTH   EZB.TN3270.sysname.tcpname.PORTnnnnn  UACC(NONE)
  ```

- Permit the user ID associated with TCP to the port profile:

  ```
  PERMIT EZB.TN3270.sysname.tcpname.PORTnnnnn CL(SERVAUTH)   ID(tcpuserid)  ACCESS(READ)
  ```

- Ensure the SERVAUTH class is RACLISTed. If it is not, RACLIST it:

  ```
  SETROPTS RACLIST(SERVAUTH)
  ```

- Refresh the in storage copy before using:

  ```
  SETROPTS RACLIST(SERVAUTH) REFRESH
  ```

Refer to *z/OS SecureWay Security Server RACF Security Administrator's Guide* for detailed information on RACF command syntax and options.

If only the first level of client authentication is desired, CLIENTAUTH SSLCERT should be specified. If the additional validation associated with the second or third level of support is wanted, CLIENTAUTH SAFCERT should be specified.

This function does not alter how RESTRICTAPPL or solicitor panel is processed.

Configuring the TN3270 server to support SSL connections: To implement SSL connections, TCP must have APF authorized access to the System SSL DLLs. The System SSL DLLs are located in hlq.SGSKLOAD by default. System SSL uses the C runtime library (SCEERUN) and the C/C++ IBM Open class library (SCLBDLL) which must also be accessible to TCP. To access these libraries, either add them to the linklist or specify them in the TCP procedure's STEPLIB. If accessed via the linklist, the linklist must be authorized (LNKAUTH=LNKLST specified in the IEASYSxx parmlib member) or the libraries explicitly APF authorized. If accessed via a STEPLIB, the libraries must be APF authorized and DISP=SHR specified. The TCP/IP profile must also be updated. An overview of the SSL related profile parameters follows. For a detailed description of the parameters, refer to *z/OS Communications Server: IP Configuration Reference*. *SecureWay Communications Server for OS/390 V2R10 TCP/IP: Guide to Enhancements* also provides information on configuring the TN3270 server for SSL connections.

The two essential parameters that must be specified are:

SECUREPORT
: All SSL enabled TN3270 ports must be defined by specifying a TELNETPARMS block for each port. The SECUREPORT port designation statement in the TELNETPARMS block indicates the port is capable of handling SSL connections.

KEYRING

As mentioned in the overview section, a server certificate is required for server authentication process defined by the SSL protocol. This certificate is stored in a keyring. The keyring type and location is specified in the KEYRING statement. Only one keyring can be used by the TN3270 server.

The keyring can be defined in the TELNETGLOBALS block. This is the preferred definition method since it ensures that the same keyring has been defined for all SECUREPORTs.

The keyring can also be specified in each of the TELNETPARMS blocks. However, if this method is used, the installation must ensure that the same keyring type and file is specified for each SECUREPORT. If the keyrings are not the same, the port definition will be rejected.

The keyring defined in the TELNETGLOBALS block overrides any keyring definitions specified in the TELNETPARMS block.

Optional SSL Related Parameters: These parameters can only be specified for SECUREPORTs. They can be specified in the TELNETPARMS block or the PARMSGROUP block. If specified in the PARMSGROUP block, they apply only to the connections mapped to the PARMSGROUP block by the PARMSMAP statement and override the parameters specified in the TELNETPARMS block. The parameters specified in the TELNETPARMS block, apply to any connection for the port that is not overridden by a PARMSGROUP definition.

The ENCRYPTION/ENDENCRYPTION Block

If the ENCRYPT/ENDENCRYPT block is specified, only the encryption algorithms included in the block are available for use. See the IP Configuration Reference for the encryption algorithms that can be specified.

The following are some reasons an installation might use this parameter:

- The applications supported on this port require high level of security and the installation wants all data encrypted using a particular encryption algorithm
- Certain connections are local and the installation does not require encryption for local clients. NULL encryption can be specified for this subset of connections.

If this parameter is not specified, any encryption algorithm supported by the installed level of System SSL is available for use.

CLIENTAUTH

If this parameter is specified, the client must send a client certificate to the server. The level of validation done depends on the option specified.

Valid CLIENTAUTH options are:

SSLCERT

To pass authentication, the Certificate Authority (CA) that signed the client certificate must be considered trusted by the server (that is, a certificate for the CA that issued the client certificate is listed as trusted in the server's keyring).

SAFCERT

Does the checking provided by SSLCERT plus verify the certificate has been registered with RACF (or other SAF compliant security product that

support certificate registration)? Additionally, if the SERVAUTH RACF class is active and a RACF resource in the format EZB.TN3270.sysname.tcpname.PORTnnnnn has been defined for the port, the connection is allowed only if the user ID associated with the client certificate has READ access to the RACF resource.

If this parameter is not specified, a client certificate is not requested during the SSL handshake and no certificate based client authentication is done.

CONNTYPE

If CONNTYPE is not specified, all connections for the SECUREPORT will use the SSL protocol (i.e. defaults to CONNTYPE SECURE).

Valid CONNTYPE options are:

SECURE
Indicates that the SSL handshake will be used to start the SSL connection. If the client does not start the handshake within the time specified by SSLTIMEOUT, an attempt will be made to do a negotiated SSL handshake (as defined by the IETF TLS-based Telnet Security Draft); if the client rejects SSL, the connection will be closed.

NEGTSECURE
Indicates the client supports the IETF TLS-based Telnet Security Draft. A TN3270 negotiation with the client first determines if the client is willing to enter into a secure connection. If the client agrees, an SSL handshake is started and SSL protocols will be used for all subsequent communication. If the client rejects SSL, the connection will be closed.

If an installation knows that the TN3270 SSL clients connecting into the port are using the protocol defined by the TLS-based Telnet Security Draft, an installation should consider using this option. With this option the SSL handshake is not attempted until after a positive response to the TN3270 DO STARTTLS IAC is received. This avoids the timeout delay that can occur when an SSL handshake is immediately started (as done with CONNTYPE SECURE) but the client is expecting the protocol used by the TLS-based Telnet Security Draft.

BASIC
Indicates that a basic (non-SSL) connection will be used.

ANY Indicates that the client can connect in either as secure or basic. The TN3270 server will first try a standard SSL handshake. If the handshake times out, a negotiated SSL (see CONNTYPE NEGTSECURE) is attempted.

- If the client is willing to enter into a secure connection, SSL protocols will be used for all subsequent communication.
- If the client is not willing to enter into a secure connection, a basic (non-SSL) connection is used.

NONE Indicates that a client is not allowed to connect in and the connection will be closed. If this option is specified in TELNETPARMS, a PARMSMAP must cover every allowable connection and the related PARMSGROUP must specify the desired CONNTYPE.

Optional CLIENTAUTH Related Parameter: The CRLLDAPSERVER block is specified in the TELNETGLOBALS block. It defines the name or IP address and port of the Certificate Revocation List (CRL) LDAP server. If CLIENTAUTH and the

CRLLDAPSERVER have been specified, the certificate revocation list is checked during client authentication. If the client's certificate is found on the certificate revocation list, the connection is closed. Only one CRL LDAP server can be defined to the TN3270 server.

Currently use of the CRL LDAP server is only supported for client certificates issued by Vault registry. (Vault Registry is an IBM product that provides the software necessary for an entity to become a Certificate Authority. Vault Registry is not a IBM Certificate Authority).

Changes to the Keyring (name, type or contents) and the CRL LDAP server (name or location) cannot be made via a VARY OBEY while connections are active. To change the Keyring or the CRL LDAP server, all secure ports must first be stopped (V TCPIP,tcpname,T,STOP,PORT=S). VARY OBEY can then be used to bring the secure ports back on line with a new keyring or CRL LDAP server. If the CRL LDAP server is stopped or connectivity is lost, System SSL may not recognize a subsequent reconnection. This situation must be handled like the CRL LDAP server change.

Using a Port for Both Basic and SSL Connections: Prior to release 10, if a port was defined as a SECUREPORT, all connections on the port had to use the SSL protocol. Because many customers want to define a single port to handle both basic and SSL-secured connections, the type of connections allowed on a SECUREPORT now include basic as well as SSL connections. SECUREPORT now indicates that the port is capable of supporting SSL connections. The default for a SECUREPORT continues to be SSL. Customers who define a port as a SECUREPORT and do not use the CONNTYPE parameter, will still get SSL for all connections on the port (see CONNTYPE SECURE for details).

Allowing a port to use both basic and secure connections assumes that either

- The installation will allow the client to determine the connection type desired.
- A subset of the connections that should use a particular connection security type can be identified (either by IP address, host name, or linkname).

In the first case, CONNTYPE ANY can be specified in the TELNETPARMS block. If the port was defined as a SECUREPORT but the client wants a basic connection, there will be a slight delay before the USSMSG screen appears. This is because when CONNTYPE ANY is coded, the TN3270 server will first attempt an SSL handshake to ensure the client isn't requesting SSL support; it is only after the SSL handshake times out that the basic connection is assumed.

In the second case, the TELNETPARMS block should specify the default connection security type (see the CONNTYPE parameter). For connections with different connection security requirements:

- Identify the connections using IPGROUP, HNGROUP, linkname or IP address.
- Create a PARMSGROUP with the alternate definitions.
- Map the connections to the PARMSGROUP using the PARMSMAP statement.

Telnet Profile Examples: The following are sample TN3270 profile definitions for a configuration with three ports with the characteristics discussed below.

- Port 23 only allows basic (non-SSL) connections.
- Ports 992 and 1023 are enabled for SSL connections and use the keyring defined in the TELNETGLOBALS block.
- Port 992 only allows SSL connections. No SSL client authentication is requested.

- Port 1023 allows both basic and secure connections. The installation desires the following characteristics for port 1023:
  - The system administrator is at IP address 9.37.88.1 and wants the capability to choose to connect with SSL or non-SSL connections.
  - Building A and B are local and do not need SSL security. The clients in these buildings have identifiable host names. The installation only wants these clients to connect in with basic (non-SSL) connections to save the encryption overhead.
  - SSL security is desired on all other connections.
  - All SSL connections require client authentication and will use the DES or triple DES encryption algorithms.

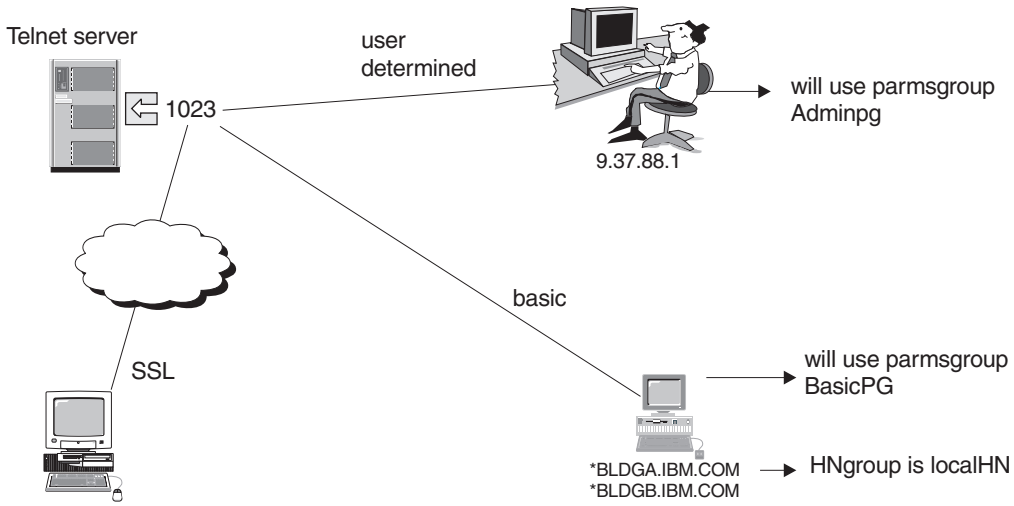


Figure 31. Port 1023 Connection Characteristics

Note: Only the definitions applicable to SSL are shown; additional parameters might be needed.

```
TELNETGLOBALS
KEYRING hfs /usr/keyring/tcpcs.kdb ;keyring used by all SECUREPORTs
ENDTELNETGLOBALS


TELNETPARMS
Port 23
; port that doesn't use SSL
...
ENDTELNETPARMS

TELNETPARMS ; port that allows only SSL connections
; no client authentication requested
; any supported encryption algorithm
SECUREPORT 992
ENDTELNETPARMS

TELNETPARMS
SECUREPORT 1023 ; port that allows SSL connections.
; note: BEGINVTAM block has PARMSGROUP that may override CONNTYPE
CONNTYPE SECURE ; SSL is default
CLIENTAUTH SSLCERT ; client certificate must be issued by a trusted CA
ENCRYPT SSL_DES_SHA SSL_3DES_SHA ENDENCRYPT ; only encrypt with DES or
;triple DES
ENDTELNETPARMS
```

```
BEGINVTAM
Port 1023
...
HNGROUP localHN *.BLDGA.IBM.COM *.BLDGB.IBM.COM ENDHNGROUP
PARMSGROUP BasicPG ; override telnetparms definitions
CONNTYPE BASIC ; don't use SSL for connections mapped to this group
ENDPARMSGROUP

PARMSGROUP AdminPG
CONNTYPE ANY ; connections mapped to this group allow any type of connection
ENDPARMSGROUP
PARMSMAP AdminPG 9.37.88.1 ; this ip address can use SSL or non SSL connections
PARMSMAP BasicPG localHN ; hosts defined in HNGROUP localHN,
;will use non -SSL connections as defined in PARMSGROUP BasicPG
ENDVTAM


BEGINVTAM
Port 992 23
...
;no PARMSGROUP defined for these ports
;TELNETPARMS definitions used for all connections
ENDVTAM
```

Creating a Keyring for the Telnet Server: To use server authentication, the Telnet server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the Telnet server to the client application.

With server authentication, the Telnet server supplies the client with the Telnet server's X.509 certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure, encrypted communication channel is established between the Telnet server and the SSL enabled client.

For server authentication to work, the Telnet server must have a private key and associated server certificate in the server's key database file.

To conduct commercial business on the Internet, you might use a widely known Certification Authority (CA), such as VeriSign, to get a high assurance server certificate. For a relatively small private network within your own enterprise or group, you can issue your own server certificates, called self-signed certificates, for your own use.

The GSKKYMAN utility or RACF Common Keyring support can be used to manage the keys and certificates needed for Telnet's SSL support.

The following high-level steps are required to enable SSL support for Telnet, with server authentication.

1. Generate the Telnet server private key and server certificate. In this release, you must decide if you will use the GSKKYMAN utility or RACF to manage your keys and certificates. If using the GSKKYMAN utility, a password file (also known as a stash file) must also be created. If using a self signed server certificate, the client's key database must also be primed with the server's certificate. See the ″Self Signed Server Certificate″ section for additional information.
2. Configure Telnet to include one or more SSL enabled ports and specify the name of the keyring created in the step above in the TELNETGLOBALS block or the TELNETPARMS block. For example:

- KEYRING hfs /usr/ssl/server.kdb (In this example, 2 files server.kdb and server.sth were created using the GSKKYMAN utility. The server's certificate is contained in the server.kdb file and designated as the default certificate.) The key database and the password stash file must reside in the same directory.
- KEYRING saf SERVERKEYRING (In this example, RACF is used to manage keys and certificates. The server certificate is connected to a keyring called SERVERKEYRING and designated as the default certificate.)

3. Restart TCP/IP or issue VARY OBEY with the updated configuration files.

If using the GSKKYMAN Utility, see "Using the GSKKYMAN Utility to Manage Keys and Certificates" on page 267. If using RACF, rather than GSKKYMAN, see "Using RACF's Common Keyring Support to Manage Keys and Certificates" on page 268.

Note: For additional information about keyring files, refer to *z/OS System Secure Sockets Layer Programming* and *z/OS SecureWay Security Server RACF Security Administrator's Guide*.

The following sections mention several certificate formats. The table below is a high level summary of the differences.

File Type	Contents	Generated by	Used by
PKCS12 format files **Notes:** 1. .p12 is usually the file extension. 2. Because this format contains the private key, the file is usually password protected. 3. We recommend only using this format where required.	• Private key • Public key • Certificate	**Notes:** 1. HOD export function, Netscape export function 2. GSKKYMAN ″Export keys to a PKCS12 file' function	1. When the HOD client specifies a client certificate to send to the server during SSL processing, it must be in this format. **Note:** The private key portion is not sent to the server. 2. To move the server certificate to another server keyring.

File Type	Contents	Generated by	Used by
Certificate files **Note:** .crt , .der commonly used as the file extension	Public key and certificate	1. HOD extract function 2. GSKKYMAN's 'Create a self-signed certificate' function	This is normally needed when a self-signed certificate is used. In this case, each self-signed certificate appears to be signed by a unique CA. Therefore, the client 's keyring (if this is a self-signed server certificate) or server's keyring (if this is a self-signed client certificate) must be primed to recognize the issuer of the self-signed certificate. This format can be used to prime a keyring with the issuer's 'CA certificate'. This format can also be used when registering a client certificate with RACF.

Using the GSKKYMAN Utility to Manage Keys and Certificates: The GSKKYMAN utility is used to create public or private key pairs and certificate requests, receive certificate requests into a keyring, and manage keys in a keyring. GSKKYMAN is a command-line utility. It prompts you for the information you need to perform a task. If you make an error, it issues a message and prompts you again for the information.

GSKKYMAN is documented in *z/OS System Secure Sockets Layer Programming*. It is recommended that you read the GSKKYMAN topics in this manual before starting to use the GSKKYMAN.

Additional information and examples can also be found in the following Redbooks:

- *SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements*
- *IBM SecureWay Host On-Demand: Enterprise Communications in the Era of Network Computing*

To run GSKKYMAN, you must have access to the z/OS Cryptographic Services message catalogs and DLLs. For example, if the z/OS Cryptographic Service DLL library is not part of the linklist concatenation, an ″export STEPLIB=hlq.SGSKLOAD″ command might be needed. For additional information, refer to *z/OS System Secure Sockets Layer Programming*.

MKKF format keyring files can be converted to GSKKYMAN format files using the -m option of the GSKKYMAN command. To do this, enter GSKKYMAN -m, and you

will be prompted for the name of the MKKF format file. For more information, refer to *z/OS Communications Server: IP Migration*.

GSKKYMAN is shipped with z/OS in System SSL as a part of the Cryptographic Services Base element of z/OS. This version of GSKKYMAN supports key sizes of 512 and 1024 bits.

Using GSKKYMAN with the Telnet Server: GSKKYMAN runs under the z/OS shell and can create several types of hfs files. For TN3270 processing, the following files are required:

- A keyring file (also known as a key database).
- A password file (also known as a stash file) which contains the password associated with the keyring file.

The keyring file and the stash file are used by the Telnet server to obtain the server's certificate and the public/private key pair used during SSL handshake processing. The Telnet server uses the stash file as the mechanism to obtain the keyring password rather than using a configuration parameter which might be accessible to a larger number of people. The stash file is created by using GSKKYMAN's 'Store encrypted database password' function on the main menu.

Security of these files is an installation responsibility. It is recommended to restrict the file access to users with superuser authority.

The TN3270 server must have read and write access to the key database and read access to the password file.

MVS files can be created from the HFS files by using the TSO OGET command with the BINARY option and can be protected using RACF. For example:

```
OGET '/tmp/telnet/mvs180.kdb' 'TCPCS6.MVS180.KDB' BINARY
OGET '/tmp/telnet/mvs180.sth' 'TCPCS6.MVS180.STH' BINARY
```

It is recommended that the MVS dsnames be the HFS filenames prefixed by one or more high-level qualifiers. The same high-level qualifier(s) must be used for both the keyring and the stash file. This ensures that the name relation used to generate the stash file from the keyring file is unchanged. Refer to *z/OS UNIX System Services Command Reference* for more information on the use of the OGET command. The MVS keyring and stash files can be used by the Telnet server.

Note: GSKKYMAN only accepts the HFS files.

GSKKYMAN allows you to enter the fully-qualified path and file name when it prompts you for a keyring, certificate request, or certificate file name. However, you should change to the path where the file should be stored before you start GSKKYMAN.

Using RACF's Common Keyring Support to Manage Keys and Certificates: In this release, RACF can be used to manage the keys and certificates normally stored in the key database. All the functions that the GSKKYMAN utility provides, are also available in the RACF support. However, because RACF can manage multiple keyrings, certificates and keyrings are added independently. A certificate is then connected to one or more keyrings.

Refer to *z/OS SecureWay Security Server RACF Security Administrator's Guide* for information about how to use RACF to manage your key database information.

All the server keyrings and certificates are stored in RACF database. There are no separate keydata base or stash files.

Before using RACF to store your key database information:

- Ensure that the DIGTCERT and DIGTRING classes are active before defining certificates or keyrings to RACF: **setropts classact(DIGTCERT DIGTRING)**. Also be sure to do a refresh after any changes. For example : **setropts raclist(digtring) refresh**.
- The RACF user ID associated with the stack must have control access to the irr.digtcert.listring resource in the FACILITY class. For example : **permit irr.digtcert.listring class(facility) id(tcpid) access(control)** where tcpid is the user ID associated with the TCP stack, not the stack name.
- Note that RACF keyring names, labels, and so on are case-sensitive. When adding the keyring name to the TCPIP profile, be sure that the correct case is used.
- The RACDCert command is used to manage most of the tasks related to keyrings and certificates. The issuer of these commands must have appropriate RACF authority to the IRR.DIGTCERT.function resource in the FACILITY class. For more information on controlling the use of the RACDCERT command, refer to the *z/OS SecureWay Security Server RACF Security Administrator's Guide*.
- Ensure that RACDCERT is defined as an authorized TSO command in the IKJTSOxx member.
- Consider RACLISTing the DIGTCERT class for best performance.

Migrating from an Existing Key Database Created by GSKKYMAN: To migrate from an existing key database (kdb) created by GSKKYMAN, each certificate that the customer has added must be individually exported and then added to the RACF database. The RACF database is already primed with some well-known Certificate Authorities (CA), so it is not necessary to migrate these CA certificates to RACF. Note however, that the well-known CAs are initially marked as NOTRUST in the RACF database and you will have to update the CA certificates that you plan to support to TRUST status.

To migrate a server certificate from your kdb created by GSKKYMAN to RACF:

1. Use GSKKYMAN to export the certificate and key to a PKCS12 format file:
   a. Open the keydata base file that you want to migrate.
   b. Select 'List/Manage keys and certificates'.
   c. Select the certificate to be exported.
   d. Select 'Export the key to a file', supply the name of the file where the certificate and key will be stored (in PKCS12 format) and enter the password to protect the file when prompted.
2. From TSO, do a binary copy of the file created in 1 above to an MVS file using the OGET command. For example, if the file name used above was mycert.p12 in the tmp directory: **OGET '/tmp/ mycert.p12' 'tcpid.mycert.p12' BINARY**.
3. Add the certificate and key to the RACF database and assign it to a user, if applicable. If this is the server certificate that is used for a particular TCP stack, assign it to the user ID associated with the stack. In this example, the user ID associated with the TCPCS stack is TCPID: **RACDCERT ID(tcpid) ADD('tcpid.mycert.p12') WITHLABEL('TCPCS-ServerCertificate') PASSWORD('mypw') TRUST**.

You will also need to create a keyring for your TN3270 server. For example:

```
RACDCERT ID(TCPid ) ADDRING(TCPCSKeyring)
```

Then connect the appropriate certificates to the keyring. For example, to connect the default server certificate that was migrated above ('TCPCS-ServerCertificate') to the 'TCPCSKeyring' that we associated with TCPID:

```
RACDCERT ID(TCPID) CONNECT( ID(TCPID) LABEL('TCPCS-ServerCertificate')
RING(TCPCSKeyring) DEFAULT )
```

To use this keyring, the TCP keyring profile data is:

```
KEYRING saf TCPCSKeyring
```

Below are some sample RACF commands that might be useful in managing your RACF keyring data. Refer to *z/OS SecureWay Security Server RACF Command Language Reference* for the full syntax and description of these commands.

- Add, delete, or list a keyring:

  ```
  racdcert ID(TCPid ) addring(TCPCSKeyring)
  racdcert ID(TCPid ) delring(SERVERKeyring)
  racdcert ID(TCPid) listring(SERVERKeyring)
  ```

- Create a self-signed server certificate call safss2 for user ID tcpid:

  ```
  racdcert ID(tcpid) gencert subjectsdn(CN('SAFSS2') ou('test') c('US'))
  TRUST size(512) withlabel('safss2')
  ```

- Connect a certificate to a keyring and make it the default:

  ```
  racdcert id(tcpid) connect(ID(tcpid) label('safss2') ring(SERVERKeyring)
  DEFAULT )
  ```

- Refresh the digtring class:

  ```
  setropts raclist(digtring) refresh
  ```

- Export the server self-signed certificate in DER format to an MVS file (to use to prime the client with the server CA certificate):

  ```
  racdcert ID(tcpid) export(label('safss2')) dsn('tcpid.safcert')
  format(certder)
  ```

- Change the Verisign Class 3 CA to trusted status and then connect it to a keyring.

  ```
  RACDCERT CERTAUTH ALTER( LABEL('Verisign Class 3 Primary CA')) TRUST

  RACDCERT ID(TCPID) CONNECT (CERTAUTH    RING(SERVERKeyring)
        LABEL('Verisign Class 3 Primary CA') USAGE(CERTAUTH)  )
  ```

RACF panels also support most of the certificate and keyring functions and can be used to perform these actions, if desired.

Self-signed Server Certificates: Normally, a server certificate should be obtained from a known Certificate Authority (CA), and the client has likely been primed with the well-known CA. However, for testing, an installation might use a self-signed server certificate. Because the clients will not know about the issuer of the self-signed server certificate, in most cases it is necessary to add the server's self-signed certificate to the client's signer certificates (HOD maintains this list in CustomizedCAs.class).

This process requires the following high-level steps:

1. Generate the server self signed certificate on the TN3270 host. This can be done with the GSKKYMAN utility or RACF.
2. Get the server's certificate to the client machine. FTP can be used for this step. If you are using newer version of HOD, this can be done directly from the HOD client (example shown below).
3. Add the server's certificate to the client's signer list. This step will differ for each client. An example using a HOD 4.0 NT client is shown below.

Detailed information for each of the above steps follows.

1. Generating the server self signed certificate.

   If using a RACF keyring:

   a. Create a self-signed server certificate using RACDCERT gencert:

      ```
      racdcert ID(tcpid) gencert subjectsdn(CN('SAFSS2') ou('test') c('US'))
      TRUST size(512) withlabel('safss2')
      ```

   b. Use RACDCERT Connect to connect the certificate to a keyring and make it the default. This example assumes a keyring called SERVERKeyring already has been created:

      ```
      racdcert id(tcpid) connect(ID(tcpid) label('safss2') ring(SERVERKeyring)  DEFAULT )
      ```

   c. If using FTP to send the server certificate information to the client, use RACDCERT Export to export the server self-signed certificate in DER format to an MVS file:

      ```
      racdcert ID(tcpid) export(label('safss2')) dsn('tcpid.safcert') format(certder)
      ```

   If using a GSKKYMAN keyring:

   a. Open your keyring file and select 'Create a self-signed certificate'.
   b. Specify Version 3, label, key size, and certificate information when requested.
   c. Set the key as the default in your key database.
   d. Save the certificate to a file, select binary format (the certificate will be saved in binary DER format).

   The following is a sample of GSKKYMAN output for creating a self-signed certificate. GSKKYMAN's default action appears in the brackets.

   a. Enter version number of the certificate to be created (1, 2, or 3) [3]: 3
   b. Enter a label for this key.....> selfsignedcert.
   c. Select desired key size from the following options (512): 1: 512 2: 1024
   d. Enter the number corresponding to the key size you want: 1
   e. Enter certificate subject name fields in the following.

      ```
      Common Name (required).............> test server certificate
      Organization (required)............> dev
      Organization Unit (optional).......>
      City/Locality (optional)...........>
      State/Province (optional)..........>
      Country Name (required 2 characters)..> US
      ```

   f. Enter number of valid days for the certificate [365]:
   g. Do you want to set the key as the default in your key database? (1 = yes, 0 = no) [1]: 1
   h. Do you want to save the certificate to a file? (1 = yes, 0 = no) [1]:
   i. Should the certificate binary data or Base64 encoded ASCII data be saved? (1 = A SCII, 2 = binary) [1]: 2
   j. Enter certificate file name or press ENTER for ″cert.crt″: ss-servercert.crt

      The following message is displayed:`Please wait while self-signed certificate is created...`

   To pick up the new default server certificates, restart TCP/IP or stop all secure ports and issue a VARY OBEY command to bring the secure ports back online.

2. Get the server certificate information to the client machine.

   - FTP can be used to ship the server certificate file create in step 1 to the client. The steps above create a binary format file and the file should be FTPed with the binary FTP option.

- The newer versions of HOD allow you to extract the server information directly from the HOD client window and eliminate the need to FTP the server certificate to the clients. The following is an example of using this method. Once the server side has been configured for the secure port and the port is active:
  - Setup your hod client to connect into the secure port and try the connection.
  - If the connection fails with a 662 (indicating the 'server presented a certificate that was not trusted'), you do not have the CA certificate for the server in you client's keyring.
    - From the client window, select communication from the action bar, then select security. Information for the server certificate should be displayed:

Figure 32. Security Information

    - Select extract and indicate binary format and where to store the certificate, then click OK.

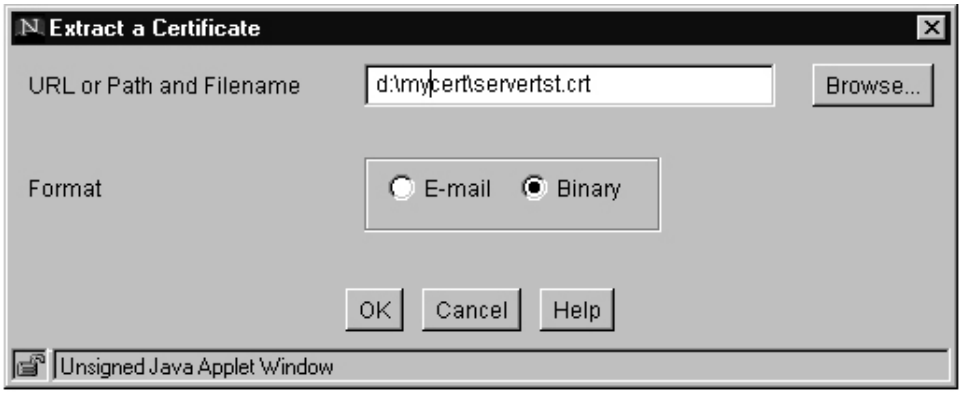


Figure 33. Extract a Certificate

If the action was successful, the following is displayed, which indicates that the server certificate information is on your client system.

Figure 34. Certificate was Extracted

3. Add the self-signed server certificate to HOD's CustomizedCAs:
   - On the HOD client, go to HOD's Certificate Management panels. To do this, select Start, Programs, IBM Host On-Demand, Administration, Certificate Management.
   - Open the CustomizedCAs.class file If customized CA certificates have previously been added to HOD, or if this is the first customized CA, create a new class file by:
     – Selecting File, then New.
     – Click on the Key Database Type arrow and select SSLight key database class. This automatically fills in the required filename and path

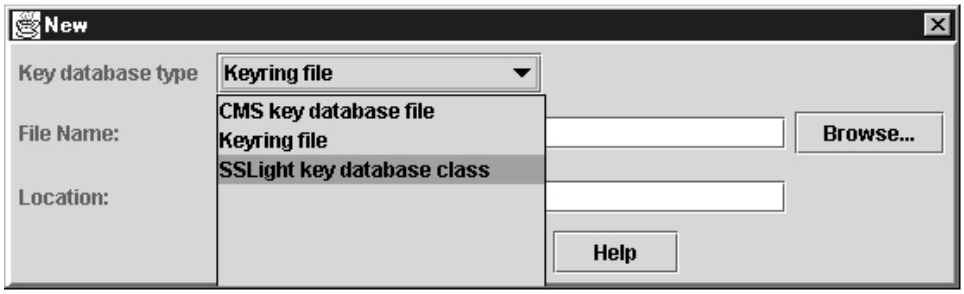


Figure 35. Creating a New CustomizedCAs.class

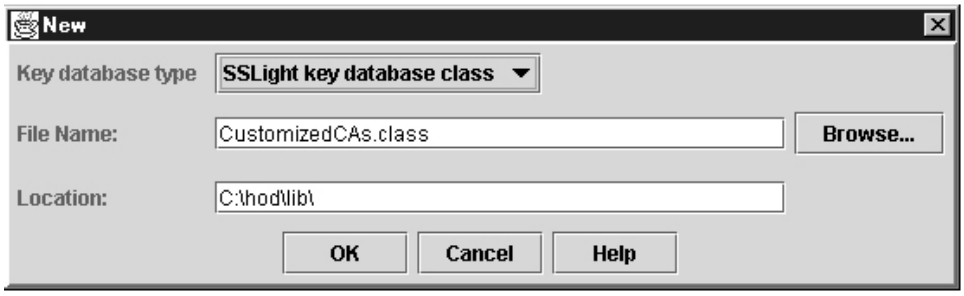


Figure 36. Default Location Displayed

     – Click OK. The *Signed Certificates* window is displayed.
     – Add the server certificate information:
       - Select ADD. The *Add CA's Certificate from a File* window is displayed.
       - Select data type 'Binary DER data'.

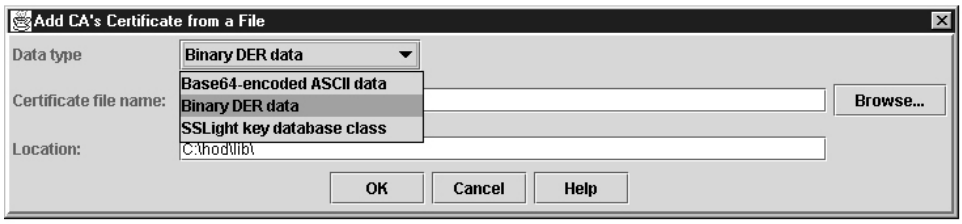


Figure 37. Add CA's Certificate From a File

– Specify the path and name of the binary certificate file that was FTPed from the TN3270 server or the file extracted using the HOD client window above. Click on the OK button to complete the add.

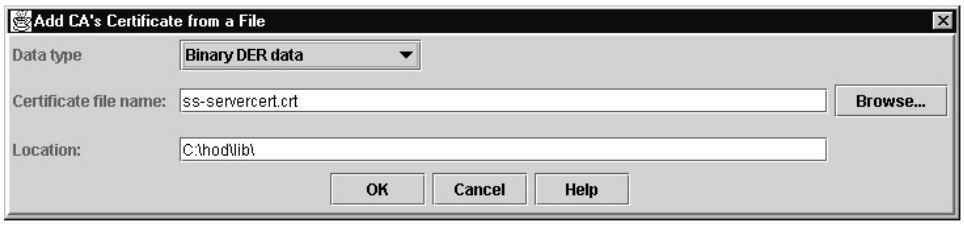


Figure 38. Add CA's Certificate From a File — Continued

– Close the CustomizedCAs.class after completing the ADD.

4. Restart HOD to pick up the updated CustomizedCAs.class.

Self-signed Client Certificates: You may have elected to implement Client Authentication at the TN3270 server by specifying either CLIENTAUTH SSLCERT or CLIENTAUTH SAFCERT in the TELNETPARMS of the Secure Port.

Normally, a client certificate should be obtained from a known Certificate Authority (CA). The Certificate Authority's root certificate needs to be included in the TN3270 server's key data base as a trusted authority in order for the client's certificate to pass the SSL protocol's client authentication process. If the client certificate has been issued by a real CA, the client certificate need not reside in the server's key database. If an installation uses self-signed client certificates for testing purposes, each certificate appears to be issued by a unique CA. Therefore, the self-signed client certificate must be added to the server's key database as a CA.

If you also want verification that the client certificate is registered with your security product (CLIENTAUTH SAFCERT specified), the client certificate must reside in the security product's database (using the RACDCERT command with the ADD option is one way to add the client certificate to RACF). Refer to *z/OS System Secure Sockets Layer Programming* for more information on using RACF to store certificates. If the installation is using self-signed client certificates and requesting verification that the client certificate is registered with security product (CLIENTAUTH SAFCERT), the client certificate must reside both in the server's key database (as a CA) and in RACF.

Steps to create a self-signed client certificate vary depending on the source of the client certificate. The example below shows the process when a client certificate created by HOD's certificate management utility is used.

Self-signed Client Certificate Created by HODs Key Management Utility: See HOD's online documentation for additional details. This sample uses a locally installed HOD V4 client on an NT system.

1. On the HOD client, go to HOD's Certificate Management panels (go to Start, Programs, Host On Demand, Administration, Certificate Management.) and open up the key database by selecting the open icon. Usually a key database will exist. If you have never used the key database, it might have a default password (usually ncod - the help menu should contain help information that specifies the default password for your system) or you can select the new option. If new is selected, the correct path and file name will normally be filled in by HOD. Do not change this file name. The HOD key database is normally in HOD's bin subdirectory and named HODClientKeyDb.kdb.

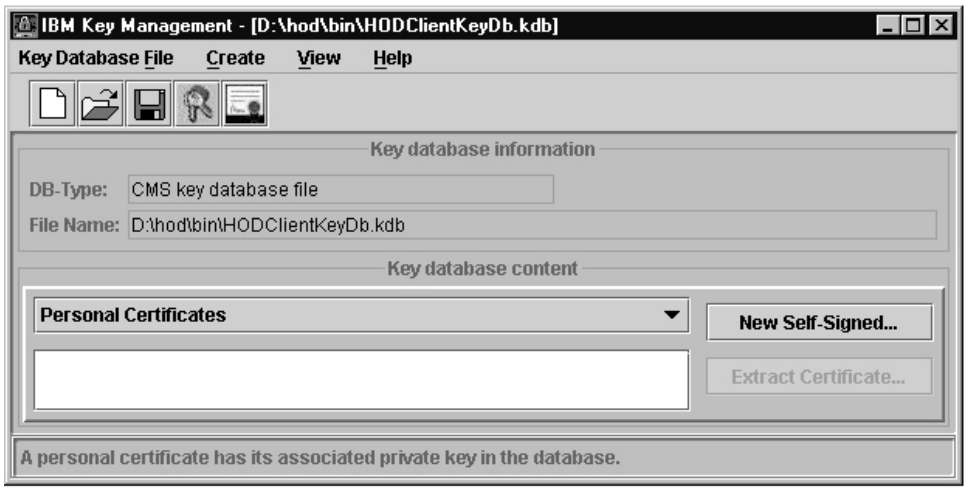


Figure 39. IBM Keys Management

   If *Personal Certificates* is not displayed, click the drop-down list arrow and select *Personal Certificates* from the pull-down list.

2. Create a self-signed personal certificate by selecting the *New Self-Signed* button. The *Create New Self-Signed Certificate* screen is displayed.

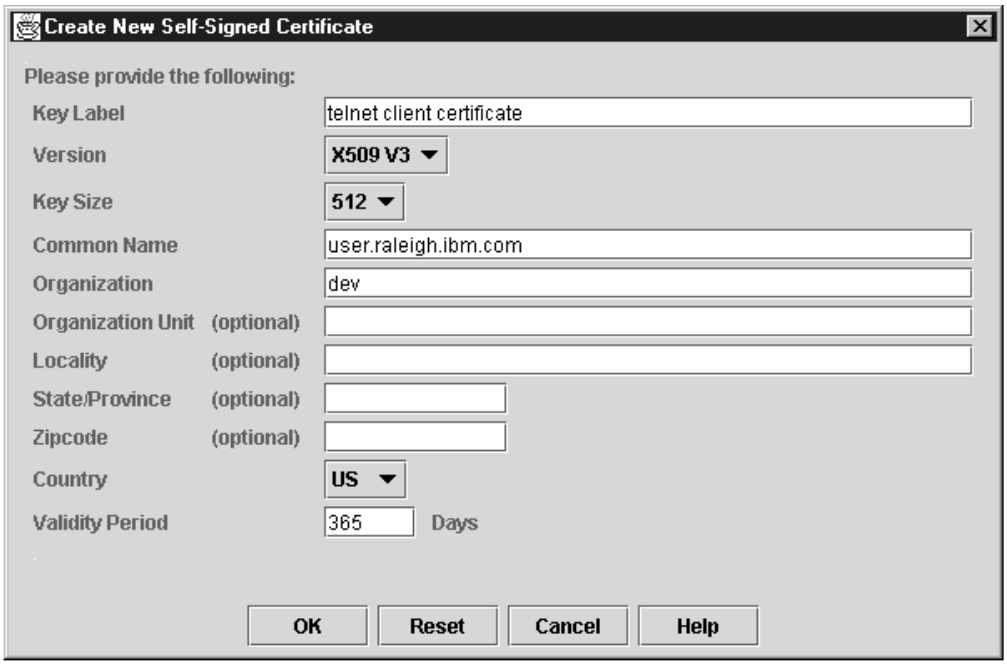


Figure 40. Create New Self-Signed Certificates

Fill in the requested information and then click OK. The new certificates will now be in the personal certificates list.

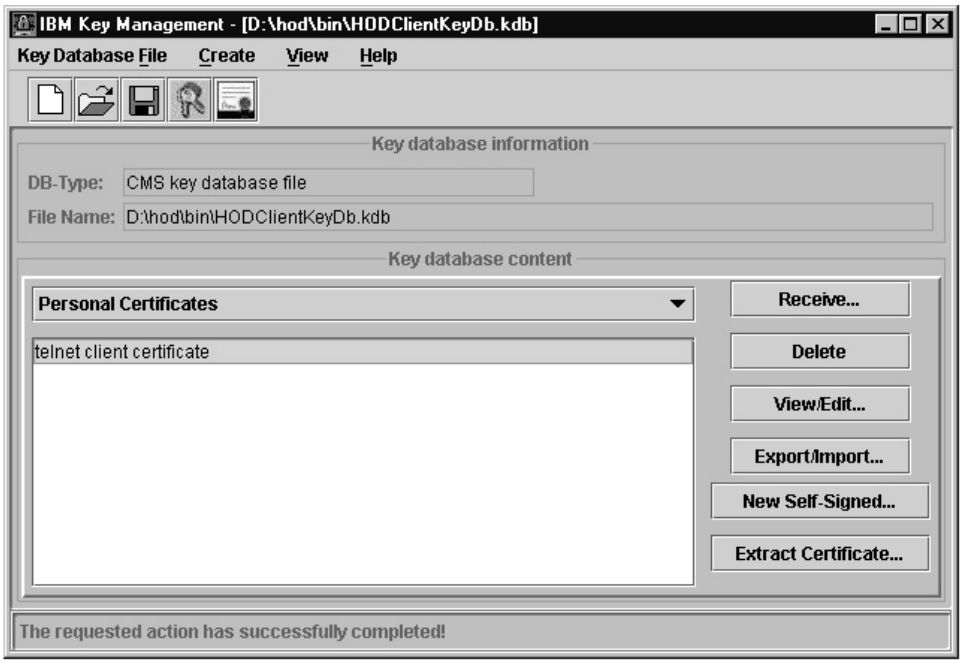


Figure 41. IBM Key Management

3. Use the export function by selecting the Export/Import button to create a PKCS12 file. This is the file that the HOD client will use.

   Specify the path and file where the exported PKCS12 file will be stored and click OK. Enter a password to protect the file when prompted.

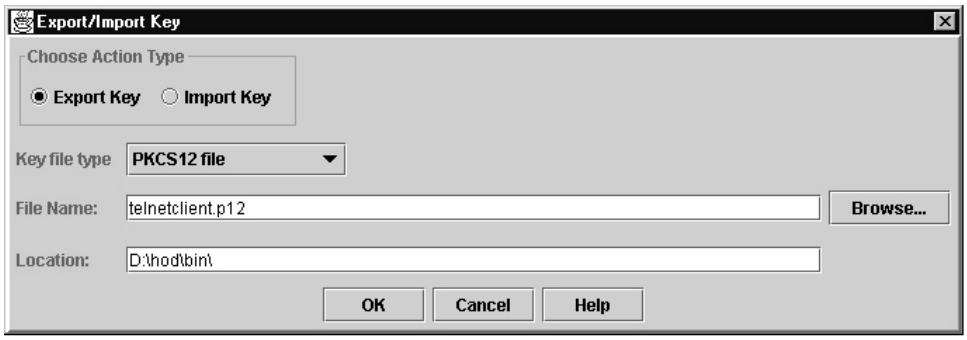


Figure 42. Export/Import Key

4. Create the certificate file that will be used to prime the server's keyring with the CA for the self-signed client certificate.

   Use the Extract Certificate function from the panel shown in Figure 41 on page 276 to create a binary DER data file. This file will have the format filename.der.

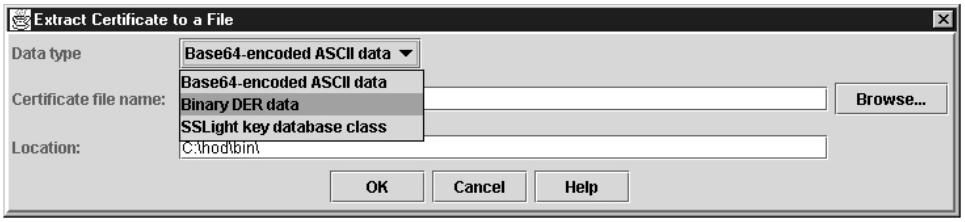


Figure 43. Extract Certificate to a File

   Specify the path and file where the exported binary DER file will be stored and click on OK.

5. FTP the binary DER data file to the TN3270 server using FTP's binary option. If using RACF for the keyring or CLIENTAUTH SAFCERT, an MVS file will need to be created. If FTP created an hfs file, use the OGET command to create an MVS file:

   OGET 'hfs-path-and-file' 'mvs-file-name' BINARY

6. On the TN3270 server, add the client CA certificate to the keyring:
   - If the server's keyring was created by GSKKYMAN, use GSKKYMAN's 'Store a CA certificate' option to obtain the client CA from the binary DER file.
   - If the server's keyring was created by RACF:
     - Use the **RACDCERT Add** command to register the client's certificate and associate it with a user ID. In this example, the binary DER client certificate has been stored in a MVS file named 'SSCLNTCERT.USER2.DER' and is associate with the RACF user ID USER2 and given a label 'CLNTCERT_USER2':

       RACDCERT ID(USER2) ADD('SSCLNTCERT.USER2.DER') WITHLABEL('CLNTCERT_USER2') TRUST

     - Use the **RACDCERT Connect** command to connect the client certificate to the keyring as a CA. In this example, the user ID associated with the TCP stack is TCPID and the keyring name used by the stack is TN3270KR:

RACDCERT ID(TCPID) CONNECT (ID(USER2) RING(TN3270KR) -
LABEL('CLNTCERT_USER2') USAGE(CERTAUTH))

7. If using the CLIENTAUTH SAFCERT, use the client binary DER format file as the source for manually registering the client certificate to the SAF product. If your keyring is a RACF keyring, this step was done above. If server keyring was created by GSKKYMAN, add the client certificate to RACF using the **RACDCERT Add** command. For example:

   ```
   RACDCERT ID(USER2) ADD('SSCLNTCERT.USER2.DER') WITHLABEL('CLNTCERT_USER2') TRUST
   ```

8. If RACF certificates were added, refresh the DIGTCERT and DIGTRING class:

   ```
   SETROPTS RACLIST (DIGTRING DIGTCERT) REFRESH
   ```

9. Restart HOD and TCP to pick up the certificates that have been added to the respective key databases. If it is not convenient to stop TCP, the updated server key database can also be picked up by stopping all secureports (V TCPIP,stackname,T,STOP,PORT=S) and then issuing a VARY OBEY to bring the secureports back on line.
10. When you start a session with HOD to a TN3270 port that requires a client certificate, HOD will display a panel that requests the client certificate file and password. The pkcs12 file created in step 3, should be specified.

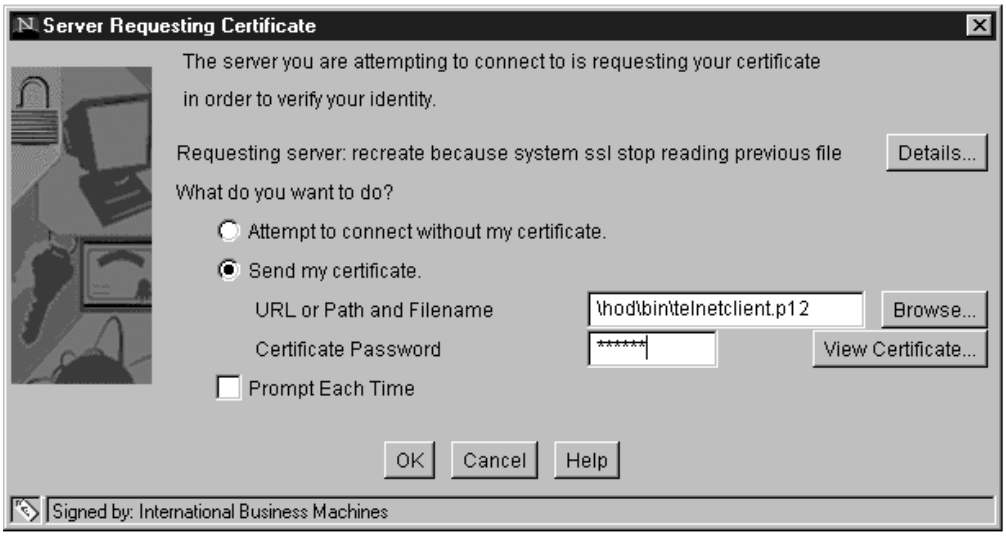


Figure 44. HOD Connection Using a Client Certificate

Note: If using HOD and you are connecting to a port that requires a client certificate, the security properties for the connection must indicate that a certificate should be sent. The following example shows the HOD Security properties screen.

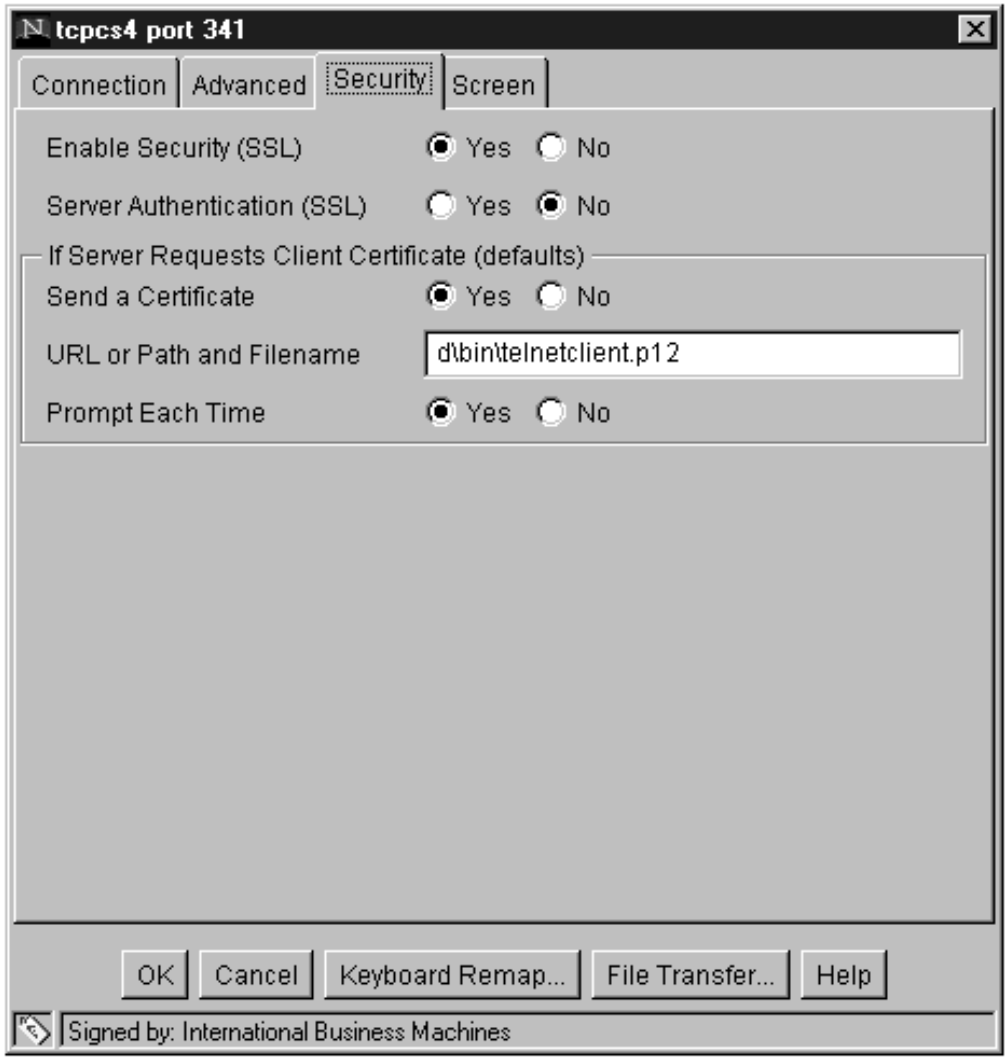


Figure 45. HOD Security Properties

Using the Telnet Solicitor or USS Logon Panel

There are three ways for an end user to establish a session with a host application.

- The client is mapped to a DEFAULTAPPL or LUMAP-DEFAPPL application
- A Telnet generated Solicitor Logon Panel requests an application name
- A USSMSG10 panel requests an application name

This section describes the Telnet Solicitor Panel and Telnet Unformatted System Services (USS) support. All information needed to establish a session can be entered on the Telnet Solicitor Panel. However, Telnet is often used as the primary method of connecting to the SNA mainframe environment. SNA end users are accustomed to entering abbreviated logon commands, altering logmodes, and entering user data from SNA terminals. For ease of migration, Telnet simulates SNA USS processing very closely. This simulation extends to being able to use the same assembled USS tables that are used by VTAM. VTAM-only character substitutions are ignored by Telnet and Telnet-only character substitutions are ignored by VTAM. Blanks are used in their place. To further extend the simulation of SNA terminals, Telnet also supports a portion of the INTERPRET table function.

Using the Telnet Solicitor Logon Panel

Telnet sends a Solicitor Panel to the end user if one of the following is true:

- No DEFAULTAPPL, LINEMODEAPPL, USSTCP, or LUMAP-DEFAPPL mappings match the Client Identifier
- A RESTRICTAPPL was requested

Below is an example of the Telnet Solicitor Panel:

```
Enter Your Userid:
Password:                          New Password:
Application:
```

Initial cursor placement can be specified. Where initial placement should be depends on client macros used and end user preferences. The OLDSOLICITOR statement in TELNETPARMS is used to implement this choice. The default cursor position is on the 'Application:' field. If OLDSOLICITOR is coded, the cursor is positioned on the 'Enter Your Userid:' field.

The userid field is used in conjunction with an application name. The RESTRICTAPPL statement defines valid user IDs for a particular application name. Once the user ID is validated and a Password is obtained, Telnet submits the user ID/Password pair for authorization to a security program such as RACF. The user ID/Password check authorizes the client to connect to the application through Telnet. The application itself might also ask for a user ID/Password pair that can be completely different than the pair entered at the Telnet Solicitor Panel. The user ID/Password pair entered at the Telnet Solicitor Panel is not in any way passed to the host application. The user ID/Password pair is solicited only after an application name is entered on the Solicitor (or USSMSG10) Panel. If a second application is reached through the original application using CLSDST PASS, the second application is verified by Telnet. However, if the second application is a RESTRICTAPPL Telnet does not solicit a new user ID/Password pair. The original user ID/Password pair is used. Therefore, if a logon manager application is used to access any RESTRICTAPPL applications, the best solution is to make the logon manager a RESTRICTAPPL with a wildcard userid. In addition to satisfying RESTRICTAPPL, there are other times when an end user might want to use the user ID/Password fields. For example, the user ID/Password pair can be used to change the current password.

Using the Telnet USS and INTERPRET Support

The Telnet USS function provides the end user with a USSMSG10 Logon Panel similar to the logon panel used by native terminals. The Telnet USS function supports sending USSMSGs to the client, receiving and parsing USSCMDs from the client, and using a translation table defined in the USS table. USSCMD parsing also includes checking for INTERPRET table entries that might provide more function than USS tables alone can provide. Sample USS and INTERPRET tables are in TCP/IP data sets hlq.SEZAINST(EZBTPUST) and hlq.SEZAINST(EZBTPINT), respectively. The USS sample has been assembled, linked, and loaded into the product data set. The tables can be used by coding the USSTCP and INTERPTCP statements in BEGINVTAM. For example, the statements below will map the sample tables to the client at IP address 1.1.1.1. See "LU Assignment– Objects, Client Identifiers, Mapping Statements" on page 235 for mapping details.

```
USSTCP   EZBTPUST  1.1.1.1
INTERPTCP   EZBTPINT  1.1.1.1
```

A new table can be created at any time and link-edited. Customized USS and INTERPRET tables can be created to change messages, commands, and

translation tables. For example, messages can be changed to have non-English text or to have different syntax. Commands can be changed to accept different syntax or to have different default values. An OBEYFILE command will cause Telnet to load the new table with the new profile being processed. Any new connection using the new profile will be assigned the new table. Telnet also supports dynamic updating of same-name USS or INTERPRET tables. The OBEYFILE adds the new version of the table to the new profile. New connections use the new copy associated with the new profile while old connections continue to use the old copy associated with the old profile.

USS Table Customization: Customized USS tables are used by both VTAM and Telnet, with any product-specific character substitutions converted to blanks. For example, @@SSCPNM is blank for Telnet and @@PRT is blank for VTAM. The tables must reside in a data set that is in the system's linklist or is in the STEPLIB statement of the TCP/IP start-up procedure. Any changes to a Telnet USS table should be made with supplementary user-defined USS tables. The IBM-supplied USS table should not be changed as it provides a good example of coding most commands and messages. Telnet loads the first table found with the name EZBTPUST and defines it as the default USS table. If this table is not found, there is no default USS table. Whether or not a default USS table should be included depends on the desired message output. When writing a USS Message, Telnet searches the USS table mapped to the client first. If the message does not exist in the mapped table, Telnet searches the default table. If the message does not exist in the default table, Telnet writes USSMSG14. If no default table exists, Telnet generates a USSMSG14. The end user can get back to the USSMSG10 from any message by pressing the CLEAR key. The default table does not affect the USS Commands. The command entered must be in the mapped table or it is not recognized.

Creating a USS Table: The following macro instructions are used to create the USS table. Telnet USS function supports almost all VTAM session-level USS message and command definitions. Refer to *z/OS Communications Server: IP Configuration Reference* for macro details.

- USSTAB - indicates the beginning of the USS table.
- USSCMD - defines commands accepted by the Telnet server.
- USSPARM - defines each operand or positional parameter that can be specified on the USSCMD macro instruction. It also defines default values for the operand or positional parameter. Multiple USSPARM macro instructions can be associated with a USSCMD macro instruction. For each operand or positional parameter code a USSPARM macro instruction.
- USSMSG - defines messages sent from the Telnet server.
- USSEND - indicates the end of the USS table.

Below are some of the more common rules to consider when coding a new USS table. Also, refer to the sample found in hlq.SEZAINST(EZBTPUST) as a guide. The following section discusses general table rules.

- If a DEFAULTAPPL or LUMAP-DEFAPPL application is mapped at the same Client Identifier level as a USS table, the USS table will only be used to return error messages and optionally after the first session logoff. A default with FIRSTONLY or LOGAPPL will send a USSMSG10 after the first session logoff. A default without the FIRSTONLY or LOGAPPL parameter will cause the connection to redrive to the default application every time, even after a successful logon and logoff.

- Only the 3270 data stream is supported. Refer to *3270 Data Stream Programmer's Reference* for more information.
- If a user-defined table is coded as part of another module, code an assembler EXTRN definition statement for the table name in that module so the table will be known externally and can be accessed by other modules.

Below are message related rules.

- USSMSGs must contain the 3270 data stream write control characters (WCCs).
- All character substitutions (@@'s) substitute the same number of fields. Any character substitution that is VTAM-specific will be translated to blanks. If the substituted value is smaller, the field is padded to the right with blanks. The parameter LUNAME or SCAN must be coded on the USSMSG macro instruction for Telnet to perform character substitutions.
- Telnet supports multiple USSPARMs with the DATA keyword. This method can be used to pass multiple data parameters to the host application. For example, two DATA USSPARMs allow the end user to type 'TSO USER1 PROC001' and have both the user ID and the Procname passed to TSO as data.

Below are command related rules.

- LOGON command format

  PL1 - logon applid(tso) logmode(snx32702) data(user1)

  BAL - logon applid=tso,logmode=snx32702,data=user1
- Any application defined in a USSCMD macro instruction must also be specified on either an ALLOWAPPL or a RESTRICTAPPL statement in the Telnet Profile.
- If the USS Command rules in *z/OS Communications Server: IP Configuration Reference* cannot be followed, use an interpret table to convert the character-coded command into a formatted SNA request.

INTERPRET Table Customization: The standard Telnet USS logon support should meet the needs of most installations. However, Telnet does support limited INTERPRET table function if special circumstances require accepting a sequence of characters outside the normal USS command format. For example, the end user might want to enter logon data that includes blanks. The INTERPRET table defines all entered data, including blanks, as a USSPARM DATA entry. The PL1 USSCMD format treats each blank as a parameter delimiter and cannot properly process a variable number of blanks. The INTERPRET table character sequences are scanned whenever the client is mapped to both a USS table and an INTERPRET table. Both must be mapped because the INTERPRET function is a subset of the USS function. INTERPRET is not a stand-alone function. The sample INTERPRET table found in hlq.SEZAINST(EZBTPINT) is not assembled and linked, and it is not loaded into Telnet as a default INTERPRET table. The table must be assembled, linked, loaded, and mapped directly in the Telnet profile to be used.

Creating an INTERPRET Table: Telnet INTERPRET function supports a limited subset of the VTAM INTERPRET definitions. Refer to *z/OS Communications Server: IP Configuration Reference* for macro details. The following macro instructions are used to create an INTERPRET table:

- INTTAB - indicates the beginning of the INTERPRET table.
- LOGCHAR - defines a single logon message and name of an application program.
- ENDINTAB - indicates the end of the INTERPRET table.

Below are some of the more common rules to consider when coding a new INTERPRET table. Also, refer to the sample found in hlq.SEZAINST(EZBTPINT) as a guide.

- The LOGCHAR APPLID= only supports APPLICID. ROUTINE and USERVAR are not supported by Telnet.
- Code the most restrictive, or longest, LOGCHAR SEQNCE values first. Otherwise, unexpected matches can occur. The table is scanned from top to bottom until a match is found whether or not it is the most exact match. For example, assume sequence 'LOGA' is assigned APPL1 and any other 'LOG' sequence is assigned APPL2. If sequence 'LOG' is before 'LOGA', entry 'LOGA' will never be found even when the end user enters 'LOGA' because entry 'LOG' will be the first match. All sessions will go to APPL2. The problem is corrected by putting 'LOGA' before 'LOG' in the table.

Assemble, Link, and Load a Table: Use the sample JCL in hlq.SEZAINST(EZBUSJCL). In the sample, the USS table is in USER1.TABLES(USSTEST). It must be assembled and link-edited as a non-executable module into the system's linklist or into a library concatenated as a STEPLIB in the TCP/IP start-up procedure. In the sample, the table is link-edited into USER1.LINKLIB(USSTEST). The same procedure can be used for the INTERPRET table. Simply change the name of the input file source and the link-edit target member. The VTAM USS and INTERPRET macros used for the assemble can be found in hlq.SISTMAC1.

Connection and Session Persistence

Connection and Session persistence is primarily a terminal connection function. Printer persistence options are limited to using the PRTINACTIVE timer or disconnecting the connection. Otherwise, the printer connection stays active and the ACB stays open. Telnet provides many options with regard to terminal connection and session persistence during all phases of a connection. Without any persistence options, the basic Connection->Logon->Logoff scenario is as follows.

- Negotiation - An end user connects to Telnet, the client and Telnet negotiate a connection mode, and a solicitor (or USSMSG10) panel is presented.
- Session Setup - The end user specifies an application name, Telnet negotiates a session between the application and the Telnet LU representing the client, and initial application data flows through Telnet to the client emulator.
- Session Logoff - When the end user is finished processing transactions and logs off the session, Telnet drops the connection.

Basic Persistence Topics

MSG07 and LUSESSIONPEND are the most common Telnet persistence statements used. They affect connection persistence in the following scenarios:

- Negotiation - If any problems occur during negotiation nothing can be done to keep the connection. If appropriate, Telnet will send the client an error code to help inform the client why the connection was dropped and issue a CONN DROP DEBUG message.
- Session Setup - If a problem occurs during session setup such as invalid application name, session request failure, or a BIND error, Telnet will drop the connection and issue a CONN DROP DEBUG message. If the MSG07 statement in BEGINVTAM is coded, the connection will not be dropped and an error message will be sent to the end user. A DEBUG DETAIL message is issued if the option is turned on. MSG07 function applies to any connection mode whether or not USS tables are associated with the connection. Press the CLEAR key to return to the USSMSG10 screen.

- Session Logoff - When the end user logs off a session, Telnet will drop the connection. If the end user typically logs on to another application after logging off the first application, it might be more efficient if the user were presented another solicitor (or USSMSG10) panel after logoff. This can be accomplished by coding the LUSESSIONPEND statement in BEGINVTAM. When LUSESSIONPEND is coded, the connection remains active but terminal LU ACBs are closed.

Advanced Persistence Topics

There are additional, more complex, persistence functions that Telnet supports.

- **KEEPOPEN the ACB** - In some cases the host application will initiate a session with the secondary LU. To do this, the host must issue an INQUIRE to see if the LU is active with an OPEN ACB. This INQUIRE will fail for Telnet LUs because Telnet does not open the ACB until a session request is sent from Telnet to VTAM. When the end user sees the solicitor (or USSMSG10) panel Telnet has not opened the ACB of the LU assigned to a TN3270E connection. TN3270 and LineMode connections do not have LUs assigned yet. If the KEEPOPEN parameter is coded on the LUMAP statement used by Telnet to assign an LU during Early Lookup for a terminal TN3270E connection, Telnet will open the ACB before sending the solicitor (or USSMSG10) panel. At that time an end user can either log on to an application as usual or wait for a host application to INQUIRE about the LU and initiate a session. TN3270 connections will not have an LU assigned until an application name is known. When the name is known, an LU is assigned to the connection, the ACB of the LU is opened, and a session request is issued. If profile statements define LUs uniquely to different applications, a second logon to a different application might fail. After the LU is assigned and opened, it stays assigned to the connection with the ACB open until the connection is dropped. When KEEPOPEN is mapped to a connection, the MSG07 and LUSESSIONPEND functions are in effect whether or not they were explicitly coded. When a session is ended, the connection remains and the ACB remains open. Only a client disconnect, a Telnet error, or the KEEPINACTIVE/INACTIVE timers will cause a KEEPOPEN connection to be dropped. The KEEPINACTIVE timer is used whenever the Telnet LU is not in session with a VTAM application. Otherwise, the INACTIVE timer is used.
- **Connection and Session TAKEOVER** - In some cases the route for a connection is lost without Telnet being notified. When this occurs, the end user can no longer make contact with the host application. The end user will typically disconnect the emulator and try to start another session. In many cases this does not work. For example, assume the host application is TSO and the end user is in session with TSO user ID USER1. The route is lost, so the end user disconnects and establishes a new connection over a different route. Assume a Generic connection so Telnet will assign a different LU to represent the client. When the TSO logon to USER1 is attempted, TSO will fail the logon because USER1 is still in session with the original Telnet LU. There is no way for the end user to bring down the original connection. The end user has to wait for an inactivity timer in Telnet or TSO to bring down the original connection. The end user is rejected sooner if a Specific connection is requested. The second Specific connection request will specify the original LU. Telnet will fail the request during Early Lookup indicating that the LU is already in use. The problem with both situations is that the original connection is still assumed active by Telnet.

  The TKOSPECLU statement in TELNETPARMS fixes this problem if the end user knows the Telnet LU name of the original connection. The statement name is derived from the idea of connection takeover (TKO) using the Specific LU (SPECLU) name. The Specific request is required as a security measure and it is assumed that most users of this function will have first connected using a

Specific request. When the connection request arrives, Telnet discovers the LU name is in use and suspends the new request. Telnet sends a TIMEMARK request to the original client which acts as an ″are you there″ message. The client is required to respond to the TIMEMARK. If no response is received by Telnet within the time specified on the TKOSPECLU statement, Telnet drops the original session and connection. The original LU is reserved during the drop process. Once the original session and connection are dropped, Telnet resumes processing the new request. This time the LU is not in use, only reserved for takeover purposes, and is assigned to the new takeover session. The end user is essentially starting over. The original session has been dropped allowing the end user to log on to the same TSO user ID again.

The TKOSPECLURECON statement in TELNETPARMS can be used to accomplish the same connection drop but avoids the session drop. When the original connection is dropped, the Telnet LU stays in session with the host application. The new connection is established and Telnet sends an LUSTAT to the host application indicating that Presentation Space Integrity was lost ('082B'x). Depending on the application, it will either end the session or resend the previous screen. By resending the previous screen, the end user is able save the original session and avoid the SNA session tear-down and restart process. At worst, if the application drops the session upon receipt of the LUSTAT the end user is able to immediately log on again as if TKOSPECLU were coded.

CAUTION: TKOSPECLURECON does not require the end user to reverify user authenticity to the host application. If there is a chance the connection can be taken over by an unauthorized user, TKOSPECLURECON should NOT be used.

Sometimes a takeover attempt will not complete as expected. This may be due to one of the following factors:

– The profile of the original connection defines how the original connection can be taken over.
– The new connection request must specify the LU name of the connection being taken over.
– TKOSPECLURECON will not preserve a session if the takeover is done from a different port. The ACB of the LU is associated with the port and must be closed before it can be associated with the new port. The takeover will function as a TKOSPECLU takeover.
– TKOSPECLURECON may not preserve a session if the original client connection is ended before the TKOSPECLURECON timer expires. If the close reason is TIMEMARK or INACTIVE, the session will be preserved under the assumption that the inactivity is due to a lost connection. Any other close reason will cause the takeover to function as a TKOSPECLU takeover. This is done to protect users who disconnect their client as a means of logging off their session. These sessions will not be taken over. Instead, the end user will have to issue a new logon.

Takeover is also affected by where the new client resides and how the old client responds. There are several event scenarios and results will vary.

– Event 1

  New client connects from a different IP address or Port.

  Original client responds to TIMEMARK.

  Result - Takeover will not occur in this case because the original client is still responding. The new client will receive an error indicating the LU is already in use.

– Event 2

  New client connects from a different IP address or Port.

Original client does not respond to TIMEMARK.

Result - Connection takeover will occur. If TKOSPECLURECON is coded, session takeover will occur. A likely scenario in this case is Telnet has lost connectivity to the old connections due to a failed router and new connections are using a different route. Or, the original machine lost power and has not reestablished connectivity. Or, the original machine lost power but reestablished connectivity with a different IP address.

– Event 3

New client connects from a different IP address or Port.

Original client stack responds with RESET.

Result - Connection takeover will occur. However, even if TKOSPECLURECON is coded, session takeover will not occur because Telnet handles the RESET as a client disconnect. A likely scenario in this case is a PC lost power and then regained power. The takeover request is accepted from either a different PC or the same PC using a different port. After the new connection request is accepted, Telnet sends a TIMEMARK to the original client PC stack. The PC stack does not recognize the IP-port and responds with a RESET.

– Event 4

New client connects from the same IP address and Port.

Telnet stack rejects the request.

Original client times out and sends a RESET.

Result - The original session and connection are dropped. Takeover does not occur. The new client is able to connect on retry because the original connection and session were cleaned up. A likely scenario in this case is a PC has lost power and then regained power. The same PC is used to attempt takeover. The client is assigned the same port as before the power loss and has the same IP address.

- **Queued Sessions** - Logon manager applications are very popular. Typically they are set up as a default application which sends a selection screen to the end user. Once the end user specifies the destination application, the logon manager typically issues a CLSDST macro with OPTCD=PASS to the new application. A session is started with the new application, the original application is closed with a special UNBIND sent to Telnet indicating that a new session BIND is forthcoming. Telnet receives the UNBIND and then waits for the next BIND instead of cleaning up as it would when receiving a normal UNBIND. When the end user logs off the destination application, Telnet will either redrive the default application or drop the connection depending on whether or not LUSESSIONPEND is coded.

  Many logon managers were written to support real terminals, not Telnet, and have the added function of issuing a SIMLOGON Q for the logon manager session itself immediately after issuing the CLSDST. When SIMLOGON is coded, the logon manager will request a session with the terminal LU (or Telnet LU representing the client) immediately after logoff from the destination application. This works very well for real terminals, but causes timing problems for Telnet when the logon manager is a default application. In this case both sides end up requesting a session.

  The QUEUESESSION statement in the BEGINVTAM block can be used to correct this timing problem. When coded, Telnet will NOT do normal close processing when the UNBIND from the destination application arrives. Telnet will wait for the BIND from the logon manager that is generated because of the

Queued SIMLOGON. Telnet verifies that the BIND is from the DEFAULTAPPL application. If it is not, the connection is dropped. QUEUESESSION does have drawbacks.

– If coded, all default applications become Qsession applications. Even if a default application does not issue a SIMLOGON Q, Telnet will wait for a BIND after the destination application logoff. In this case, the connection appears hung to the end user. Only a Telnet timer expiration or a client disconnect will clean up the connection.
– QUEUESESSION and LUSESSIONPEND are mutually exclusive. If any connections are not using the default application the connection will be dropped at session logoff time. LUSESSIONPEND has a higher priority if both are coded.
– Only the logon manager application can send a BIND to Telnet. If a chain of CLSDST PASSs are issued to other applications, none of the intermediate applications can issue a SIMLOGON Q. If they do, that BIND will flow to Telnet which will then drop the connection because the BIND was not for the logon manager application.

However, QUEUESESSION is a good solution for a simple setup where all connections begin with a Qsession-type logon manager.

The QSESSION parameter on the ALLOWAPPL or RESTRICTAPPL statement is a more granular and more functional solution to the Qsession issue than QUEUESESSION.

– QSESSION can be applied to one or any number of applications. Any ALLOWAPPL or RESTRICTAPPL that has QSESSION coded becomes known as a Qsession-type application by Telnet.
– LUSESSIONPEND is independent of the Qsession parameter.
– QSESSION supports up to 10 intermediate Qsession applications. Any one of them can issue a SIMLOGON Q after the CLSDST PASS and Telnet will recognize the BIND that flows to Telnet after the destination application logoff occurs. QUEUESESSION supports a BIND only from the logon manager. QSESSION supports a BIND from any of the first 10 applications in a CLSDST PASS chain.

As an example of the QSESSION parameter, assume that APPL1, APPL2, and APPL3 are each defined in VTAM with SimlogonQ LIFO (Last In, First Out). The following Telnet statements allow connections to access the applications and define which is a QSESSION application.

```
ALLOWAPPL APPL1 QSESSION
ALLOWAPPL APPL*
```

In this example, the client first logs on to APPL1 and Telnet saves the name in the first slot of the 10 slot Qsession table. A CLSDST-PASS to APPL2 is done and the APPL2 name is saved in slot two of the table. Finally, a CLSDST-PASS to APPL3 is done and the APPL3 name is saved in slot three of the table. Each CLSDST-PASS specifies to VTAM that the existing session should be queued in a LIFO arrangement. When the session with APPL3 is ended, Telnet clears the slot where APPL3 was saved and VTAM sends an APPL2 BIND to Telnet. Telnet verifies the ACB address and verifies that APPL2 is in the Qsession table. Verification succeeds and a new session is established with APPL2. Next, the APPL2 session ends. The APPL2 slot is cleared and VTAM sends an APPL1 BIND to Telnet. Verification succeeds, and a new session with APPL1 is established. When the APPL1 session is ended, the LU is cleaned up and the connection status depends on whether or not LUSESSIONPEND is coded.

Another example illustrates table manipulation. Assume the Simlogon queuing method is FIFO (First In, First Out) instead of LIFO. The same process is used to get into session with APPL3. When the APPL3 session is ended, the APPL3 slot is cleared and VTAM sends an APPL1 BIND to Telnet. Because FIFO queuing was used, the first application queued is the first application off the VTAM queue. In this case Telnet will start at the end of the Qsession table (APPL2 since APPL3 entry was just cleared) and check each slot to try to find an application name match. Eventually a name match is made with slot one. Now, all slots above slot one are cleared. A CLSDST-PASS to APPL4 will cause the APPL4 name to be put into slot 2. Because the slots for APPL2 and APPL3 were cleared earlier, Telnet will no longer accept a BIND from APPL2 after ending the session with APPL4.

The Qsession table is set up when a BIND is received for an application that is defined with the QSESSION parameter. If the first application does not have QSESSION coded and a CLSDST-PASS is issued to an application name that has QSESSION coded, it is the second application that will be in slot one of the Qsession table. When that session in slot one is ended, Telnet will clean up the LU and the connection status will depend on whether or not LUSESSIONPEND is coded.

- **DISCONNECTABLE** parameter on either the ALLOWAPPL or RESTRICTAPPL statement determines what type of session termination to send to the host application when Telnet initiates session termination. If DISCONNECTABLE is coded Telnet issues a TERMSESS UNBIND(0F). Otherwise, Telnet issues a TERMSESS UNCOND. For example, when DISCONNECTABLE is coded for the TSO application, an unexpected connection loss results in an UNBIND(0F) being sent to TSO putting it in a reconnectable state. The DISCONNECTABLE parameter has no affect on a session ended normally by the end user logging off the session. The QSESSION parameter can be coded with DISCONNECTABLE on either statement.
- **LOGAPPL and FIRSTONLY** parameters can be coded on LUMAP-DEFAPPL, DEFAULTAPPL, or LINEMODEAPPL (for the remainder of this section, LINEMODEAPPL also applies whenever DEFAULTAPPL is mentioned). Both LUMAP-DEFAPPL and DEFAULTAPPL support the LOGAPPL, FIRSTONLY, and DEFONLY parameters. See “Application Security” on page 253 for DEFONLY security implications.

  The LOGAPPL function keeps the Telnet LU active if a Request Session fails because the host application is not active. In addition, VTAM remembers the attempted Request Session and will initiate a session request to the Telnet LU on behalf of the host application when the application becomes active. When the Request Session fails, Telnet sends the client a solicitor panel or USSMSG7 screen. The end user then has the option of logging on to a different host application (if DEFONLY is not coded). When this different session is started, VTAM will drop the queued Request Session for the original session.

  What happens at session logoff depends on whether or not FIRSTONLY and LUSESSIONPEND are coded. As a general rule, if LUSESSIONPEND is not coded, the connection is dropped. If FIRSTONLY is not coded, Telnet will issue another Request Session to the default application defined by either LUMAP-DEFAPPL or DEFAULTAPPL. If FIRSTONLY is coded, Telnet will send a USSMSG10 screen or Solicitor Panel to the client.

  The following table summarizes several possible session initiation failure scenarios.

– ReqSess OK - A Request Session to the default application succeeded or a Request Session to a second application from the USSMSG07 screen succeeded.
– ReqSess Fail - A Request Session to the default application failed.
– 2nd Appl Fail - A Request Session to a second application from the USSMSG07 screen failed.

Scenario / Mapping Statement	ReqSess OK	ReqSess Fail		2nd Appl Fail	
		MSG07	No MSG07	MSG07	No MSG07
DEFAULTAPPL name LUMAP-DEFAPPL name	1	2	5	2	N/A
DEFAULTAPPL name LOGAPPL LUMAP-DEFAPPL name LOGAPPL	1	3	3	4	4

Figure 46. Session Initiation Failures Scenarios

FIRSTONLY is not a consideration because the first session has not been established.

1. In Session.
2. Send USSMSG07 or Solicitor panel to client. Close the ACB.
3. Send USSMSG07 or Solicitor panel to client. Keep ACB open, queue original Session Request in VTAM.
4. Send USSMSG07 or Solicitor panel to client. Keep ACB open, keep the queued Session Request in VTAM.
5. Drop Connection.

The following table summarizes several possible session ending scenarios. The session is ending due to the normal LOGOFF or session breakage (possibly caused by loss of the application).

- Original Application - User is in session with the original default application.
- Original Application - User is in session with the original default application.
- CLSDST from Original Appl - User is in session with a second (or later) application after issuing a CLSDST-PASS from the original application.

Scenario / Mapping Statement	Original Application		2nd Appl or CLSDST from Orig	
	Logoff	Break	Logoff	Break
DEFAULTAPPL name LUMAP-DEFAPPL name	1	1	1	1
DEFAULTAPPL name LOGAPPL LUMAP-DEFAPPL name LOGAPPL	2	1	1	1
DEFAULTAPPL name FIRSTONLY LOGAPPL LUMAP-DEFAPPL name FIRSTONLY LOGAPPL DEFAULTAPPL name FIRSTONLY LOGAPPL LUMAP-DEFAPPL name FIRSTONLY LOGAPPL	2	1	2	1

Figure 47. Session Ending Scenarios

In all cases, if LUSESSIONPEND is not coded the connection is dropped.

1. Redrive the default application.
2. Send USSMSG10 or Solicitor panel to client. Close the ACB.

Timers

Several timers are available in Telnet to control how long connections stay up. The list includes:

- INACTIVE - How long a terminal connection can be idle with no SNA data traffic before the connection is dropped.
- PRTINACTIVE - How long a printer connection can be idle with no SNA data traffic before the connection is dropped.
- KEEPINACTIVE - How long a KEEPOPEN connection can be idle with no SNA data traffic before the connection is dropped.
- SCANINTERVAL - How often Telnet runs the list of connections looking for potentially lost connections. Because of the methodology, it also determines how long Telnet will wait for a TIMEMARK response before assuming the connection is lost.
- TIMEMARK - How long a connection is active without receiving any data before Telnet sends a TIMEMARK command which acts as an ″are you there″.
- SSLTIMEOUT - How long Telnet will wait for an SSL handshake response before the request is dropped.

To facilitate these timers, Telnet records the time at which data is received from the client, received from VTAM, or sent to VTAM. Data received from the client is used by SCANINTERVAL/TIMEMARK to measure idle time on the connection. Data received from or sent to VTAM is used by the INACTIVE family of timers to measure idle time without SNA data traffic.

INACTIVE, PRTINACTIVE and KEEPINACTIVE all share one timer associated with a profile to reduce system overhead. The timer with the smallest value is used to define how often the connections are checked. The other two timers are then rounded up to the next integer multiple of the smallest timer. For example, assume KEEPINACTIVE is defined as 1800, INACTIVE is defined as 3000, and PRTINACTIVE is defined as 5400 in a profile. The Telnet timer will run every 1800 seconds. Therefore, every time the timer pops Telnet will check each KEEPOPEN connection to see if there has been any SNA data traffic in the prior 1800 seconds. If not, the connection is dropped with DEBUG SUMMARY message CONN DROP reason INACT-K. The INACTIVE timer is rounded up to 3600 seconds. Therefore, every second time the timer pops Telnet will check each terminal connection to see if there has been any SNA data traffic in the prior 3600 seconds. If not, the connection is dropped with DEBUG SUMMARY message CONN DROP reason INACT-S. The PRTINACTIVE timer remains at 5400 seconds. Therefore, every third time the timer pops Telnet will check each printer connection to see if there has been any SNA data traffic in the prior 5400 seconds. If not, the connection is dropped with DEBUG SUMMARY message CONN DROP reason INACT-P. Setting KEEPOPEN to the smallest time was done as an example. Any three of the timers could be the smallest.

SCANINTERVAL and TIMEMARK are used together to determine if a connection has been lost. Whenever data is received from the client, Telnet records the time. Telnet checks all connections at regular intervals defined by the SCANINTERVAL value. Each connection is checked to see if any data has been received from the client in the past TIMEMARK period of time. If not, a TIMEMARK command is sent to the client which acts as an ″are you there″ and Telnet remembers a TIMEMARK was sent to this client. During the next check at SCANINTERVAL time later, each connection is again checked to see if any data has been received from the client. If not, and a TIMEMARK was sent on the previous check, the connection is dropped with DEBUG SUMMARY message CONN DROP reason TIMEMARK. For example,

assume the values for SCANINTERVAL and TIMEMARK are 1800 and 10800, respectively. That means every 30 minutes all connections are checked to see if any data has been received in the last 3 hours. If not, a TIMEMARK is sent to the client. 30 minutes later Telnet checks the connections again. If the client responded to the TIMEMARK or sent in actual data of some type Telnet leaves the connection active. If nothing has been received Telnet drops the connection.

Caution must be used in setting these timers. Setting the INACTIVE family of timers or SCANINTERVAL timer too low could cause excessive CPU usage. Setting the TIMEMARK value too low could also cause excessive flooding of the network with TIMEMARK commands. For example, these timers should take into account extended breaks such as lunch. If TIMEMARK is smaller than the lunch break time, the network may be flooded with TIMEMARK commands around the lunch hour. Be aware of the default values and be sure to set appropriate values for the situation.

SSLTIMEOUT is different than the other timers. Telnet does not run this timer. The time value is passed to the SSL handshake process. If SSL does not get a response from the client within SSLTIMEOUT period of time, the handshake request fails. Telnet will then proceed to the next available connection negotiation method or drop the connection.

Telnet Diagnostics

In addition to general diagnostic tools such as CTRACE and dumps which are described in *z/OS Communications Server: IP Diagnosis*, there are Telnet specific diagnostic tools available. Profile syntax can be checked, display requests can be issued, session initiation and termination can be tracked, and error messages can be specified.

TESTMODE

The TESTMODE statement in TELNETPARMS allows a profile to be processed by Telnet but then dropped before it becomes an active profile. Using TESTMODE ensures that LU assignments, security levels, and other Telnet parameters are not compromised due to profile syntax errors.

DISPLAYS

Several displays are available which provide summary and detail information. Telnet provides Profile-related displays to present the following:

- General information such as SMF subtype, timer values, and security options.
- Mapping information such as listing groups defined, how those groups are mapped to Client Identifiers, how many LUs within a group are in use, and how many users are connected to a certain application.
- WhereUsed information such as where names or IP addresses are used within a profile.
- Connection information such as a summary list of all connections, a filtered summary list, or a detailed display of a single connection.

  The following Telnet displays are available to perform the above functions. Refer to *z/OS Communications Server: IP Configuration Reference* for detailed examples.

General Information:

- PROFILE - General profile information, such as timers and security information.
- DEVICETYPE - Summary of all device types and their associated logmodes.
- WLM - Summary of all workload manager names registered to represent Telnet.

Mapping Information:

- LUGROUP - Summary of all LUGROUP and PRTGROUP statements or detailed data of a single group including which LUs are in use.
- IPGROUP - Summary of all IPGROUP statements or detailed data of a single group including which IP addresses have active connections.
- HNGROUP - Summary of all HNGROUP statements or detailed data of a single group including which host names have active connections.
- PARMSGROUP - Summary of all PARMSGROUP statements or detailed data of a single group.
- LUMAP - Summary of all LUMAP and PRTMAP statements.
- PARMSMAP - Summary of all PARMSMAP statements.
- DEFAULTS - Summary of all DEFAULTAPPL, LINEMODEAPPL, USSTCP, and INTERPTCP mapping statements.
- APPL - Summary of all ALLOWAPPL and RESTRICTAPPL statements or detailed data of a single statement including number of sessions and application-based LU mappings if appropriate.
- LINKNAME - Summary of all link names and their IP address used on Telnet mapping statements.
- INACTLUS - Summary of all LUs that have been inactivated by the operator or by Telnet as a result of OPEN ACB problems.

WhereUsed information:

- WHEREUSED - Summary of all uses and mappings for a given name or IP address.

Connection Information:

- CONNECTION - Summary of all connections, filtered summary of all connections, or detailed data of a single connection.

The above displays give good snapshots of the state of Telnet and its connections. In addition, WHEREUSED is very useful as a lookup debug tool. The display will show where any name or IP address is used within the profile. This is particularly helpful when a connection request does not proceed as expected. For example, a client is sent a USSMSG10 screen instead of immediately getting in session with a logon manager. A WHEREUSED display of the client's IP address might show, as expected, the IP address within an IP group that is mapped to the logon manager as a default application. However, another WHEREUSED display of the client's host name might show the host name within an HN group that is mapped to a USS table. Knowing that the selection order of Client Identifiers will cause the USS table to be chosen, the administrator will realize there are conflicting mapping statements for the client and will be able to resolve the problem.

Module Information:

Telnet module storage can be displayed to verify the level of maintenance. For example, symptoms indicate a problem that has been fixed by an APAR. It is not known for sure if the fixed module is in the current version of Telnet. The module in question can be displayed using the TCP/IP display STOR command. In the example below no APAR is listed, indicating it has not been applied to the system.

```
d tcpip,,stor,mod=ezbtvxrc
  EZZ8456I TCPIP MODULE STORAGE
    EZBTVXRC LOADED AT 0A220328 IN LOAD MODULE EZBTTMST
    +0000 47F0F026 20C5E9C2 E3E5E7D9 C340F0F0 *.00..EZBTVXRC 00
```

```
   +0010 4BF0F0F5 40F0F17A F2F97AF2 F240C8E3 *.005 01.29.22 HT
   +0020 C3D7F5F0 C1000DC0 58300224 58403150 *CP50A........ ..
 EZZ8459I DISPLAY TCPIP STOR COMPLETED SUCCESSFULLY
```

SMF

SMF records are written when an end user establishes a session (SMF LOGN) and when the session is ended (SMF LOGF). Optional SMF recording is controlled using the SMFINIT and SMFTERM statements in TELNETPARMS. Data recorded includes the application name, Telnet LU name, client and host IP address and port, time of logon or logoff, and data count in and out. Combined with the SMF utility exit routine, SMF data can be used to track Telnet usage by a number of variables. Refer to *z/OS Communications Server: IP Configuration Reference* for SMF record layout.

DEBUG

The DEBUG SUMMARY statement in TELNETPARMS provides another tracking method. A summary message is written when:

- A connection request is accepted by Telnet.
- Connection negotiation is complete.
- A session is established with the host application.
- A session is dropped.
- A connection is dropped.

LU name, Connection ID, and client IP address and port are included in each message. In the example below an end user connects to port 23. The connection is negotiated as a TN3270E connection and a session with TSO is established. The session is dropped because of client disconnect (CLNTDISC) and then the connection is dropped because of client disconnect.

```
EZZ6034I TELNET CONN 00000011 LU **N/A**  ACCEPTED      23
                      IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 NEGOTIATED TN3270E
                      IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 IN SESSION TSO0001
                      IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 SESS DROP  CLNTDISC
                      IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 CONN DROP  CLNTDISC
                      IPADDR..PORT 1.12.13.14..456
```

In addition to tracking major state changes and providing key information, the statements can be used for debug purposes. For example, an end user might be attempting a connection and something is not working. The ACCEPTED, NEGOTIATED, and IN SESSION messages are major connection milestones. Using the information provided and knowing whether or not these messages are displayed can provide many clues about the connection request. The SESS DROP and CONN DROP messages give a variety of reasons about why the drop occurred. Refer to *z/OS Communications Server: IP Messages Volume 3 (EZY)* message EZZ6034I for reason details. The CONN DROP message is issued for error conditions and inactivity reasons whether or not DEBUG is coded. If DEBUG is not coded, subsequent messages with the same reason received within 15 seconds will be suppressed.

If DEBUG SUMMARY messages do not provide enough information to solve a problem, try DEBUG DETAIL. In addition to the summary messages listed above, DEBUG DETAIL will issue a message at the time of failure which displays the client IP address and port, connection ID, Telnet LU name, detecting module name, unique return code and brief explanation, and additional parameters if relevant.

Some messages will be helpful to the system administrator and others will be helpful to IBM service. Refer to *z/OS Communications Server: IP Messages Volume 3 (EZY)* message EZZ6035I for return code details.

In the examples below the end user specified an application not in the Telnet profile and then disconnected at the client emulator.

```
EZZ6035I TELNET DEBUG 1.12.13.14..456
     CONN: 00000011  LU: TCPM1001  MOD: EZBTPGLU
     RCODE: 3012-00  Application name is invalid.
     PARM1: 00000000 PARM2: 00000000 PARM3: 00000000

   EZZ6035I TELNET DEBUG 1.12.13.14..456
     CONN: 00000011  LU: TCPM1001  MOD: EZBTTRCV
     RCODE: 1001-02  Client disconnected from the connection.
     PARM1: FFFFFFFF PARM2: 00000461 PARM3: 00000000
```

The DEBUG option may cause flooding of the operator's console. Console flooding concerns can be dealt with in several ways.

- DEBUG messages are, by default, assigned to routing code 11 - the JOBLOG. The optional DEBUG parameter JOBLOG can be used for the same effect. However, the master console also receives routing code 11 messages by default. To stop the messages from going to the master console, issue ″VARY CN(01),DROUT=(11)″, which drops routing code 11 from the console. The other optional DEBUG parameter, CONSOLE, will direct the messages to the master console, routing code 2, and the teleprocessing console, routing code 8.
- If DEBUG SUMMARY messages are being used primarily for problem debug, the OBEYFILE command can be used to keep the number of messages low. Bring up Telnet initially without DEBUG coded. When a problem appears, issue an OBEYFILE for a Telnet profile that includes the DEBUG statement. Only new connections to the new profile will produce messages. Once data is obtained, issue another OBEYFILE for a Telnet profile that omits the DEBUG statement.
- If the host name or IP address of the client having the problem is known, include DEBUG in a PARMSGROUP statement. Using PARMSMAP, map that group to the client IP address or host name. Debug messages for only that client will be issued.

The MSG07 statement in BEGINVTAM is very helpful. It allows Telnet to send a message to the client indicating an error occurred and what the error was. Something simple like a mistyped application name can be corrected by the end user without additional help. Even for more difficult problems, MSG07 provides a good starting point. It is recommended that MSG07 always be coded unless there are reasons not to send error messages to the client. If MSG07 were not coded in the DEBUG DETAIL example above, the connection would have been dropped after the lookup failure. The CONN DROP message would include the return code and indicate that an error caused the connection drop. The following message would be produced whether or not DEBUG was coded because of the error condition.

```
 EZZ6034I TELNET CONN 00000011 LU TCPM1001 CONN DROP  ERR 3012
                      IPADDR..PORT 1.12.13.14..456
```

CTRACE

If a problem is not resolved using the above tools, IBM service will likely need a CTRACE with option Telnet. CTRACE, with only the Telnet option, gives very complete information about the Telnet processes. To debug almost any Telnet problem no other CTRACE option is needed. Generally, the other options simply take up space creating a trace-wrap condition more quickly. If the problem is data related, use the FULLDATATRACE statement in TELNETPARMS to trace all the data coming into and leaving Telnet rather than tracing only the first 64 bytes of

data. FULLDATATRACE might cause a trace-wrap condition more quickly so should be set only if needed. For transform problems, the DBCSTRACE statement in TELNETPARMS should be used to produce more trace entries in the SYSPRINT and TNDBCSER data sets.

Configuring the z/OS UNIX Telnet Server (otelnetd)

The z/OS UNIX Telnet server provides access to z/OS UNIX shell applications on the host using the Telnet protocol.

Installation Information

The HFS files used in the z/OS UNIX Telnet server and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file. otelnetd writes to syslog facility local1.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/otelnetd
This is a symbolic link to /usr/lpp/tcpip/sbin/otelnetd. /usr/lpp/tcpip/sbin/otelnetd is a sticky-bit file. The member of OTELNETD of hlq.SEZALINK contains the executable code for the Telnet server.

/etc/banner
This file contains a login message which will be printed to the client's screen unless the -h option is specified. This banner should be stored here.

/bin/fomtlinp
The executable file for utmp entry is stored here. This code will update the utmp entry as well as generate the child.

/etc/utmpx
This file is updated by the call to fsumoclp. It contains a list of all the users who are logged in with their associated tty.

/dev/ptypXXXX and /dev/ttypXXXX
These special device files represent pseudoterminals (ptys); they are used by the z/OS UNIX Telnet server and other programs.

Note: For information on allocating more of these files for more connections, see *z/OS UNIX System Services Planning*.

/usr/share/lib/terminfo
The descriptions of supported terminals are stored here. For more information, see *z/OS UNIX System Services Planning*.

/usr/lib/nls/msg/C/tnmsgs.cat
The message catalog used by the z/OS UNIX Telnet server is stored here.

If the message catalog does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

Note: The z/OS UNIX inetd daemon does not propagate environment variables other than PATH and TZ to its child processes, so the NLSPATH and LANG environment variables cannot be used to point to a different message catalog.

/usr/man/C/cat1/otelnetd.1
This file contains the associated manual (man) pages for the z/OS UNIX Telnet server. It provides online help for the user.

Starting, Stopping, and Administration of z/OS UNIX Telnet

The z/OS UNIX Telnet server is started by inetd for each incoming Telnet connection. When the Telnet session is complete, the z/OS UNIX Telnet server will exit. Each active Telnet session will have a separate instance of the Telnet server which will communicate with the Telnet client.

The following standards are supported:
- RFC 854 Telnet Protocol Specification
- RFC 855 Telnet Option Specification
- RFC 856 Telnet Binary Transmission
- RFC 857 Telnet Echo Option
- RFC 858 Telnet Suppress Go Ahead Option
- RFC 859 Telnet Status Option
- RFC 860 Telnet Timing Mark Option
- RFC 861 Telnet Extended Options - List Option
- RFC 885 Telnet End of Record Option
- RFC 1073 Telnet Window Size Option
- RFC 1079 Telnet Terminal Speed Option
- RFC 1091 Telnet Terminal type option
- RFC 1096 Telnet X Display Location Option
- RFC 1123 Requirements for Internet Hosts -- Application and Support
- RFC 1184 Telnet Linemode Option
- RFC 1372 Telnet Remote Flow Control Option
- RFC 1571 Telnet Environment Option Interoperability Issues
- RFC 1572 Telnet Environment Option

When an OS/390 UNIX Telnet session is started up, otelnetd sends Telnet options to the client side indicating a willingness to do the following:
- DO TERMINAL TYPE
- DO TSPEED
- DO XDISPLOC
- DO NEW-ENVIRON
- DO ENVIRON
- WILL SUPPRESS GO AHEAD
- DO ECHO
- DO LINEMODE
- DO NAWS
- WILL STATUS
- DO LFLOW
- DO TIMING-MARK

The z/OS UNIX Telnet server can enable the following options locally:
- WILL BINARY

  This option indicates that the client is willing to send 8 bits of data, rather than the normal 7 bits of network virtual terminal data.
- WILL ECHO

When the LINEMODE option is enabled, a WILL ECHO or WONT ECHO will be sent to the client to indicate the current state of terminal echoing. When terminal echo is not desired, a WILL ECHO is sent to indicate that Telnet will take care of echoing any data that needs to be echoed to the terminal, and then nothing is echoed. When terminal echo is desired, a WONT ECHO is sent to indicate that Telnet will not be doing any terminal echoing, so the client should do any terminal echoing that is needed.

- WILL LOGOUT

  When a DO LOGOUT is received, a WILL LOGOUT is sent in response and the Telnet session is shut down.
- WILL SGA

  This option indicates that it will not be sending IAC GA, the go ahead command.
- WILL STATUS

  Indicates a willingness to send the client, upon request, the current status of all Telnet options.
- WILL TIMING-MARK

  Whenever a DO TIMING-MARK is received, a WILL TIMING-MARK is the response. It is only used in kludge linemode support.

The z/OS UNIX Telnet server can enable the following options remotely:

- DO BINARY

  Sent to indicate that Telnet is willing to receive an 8-bit data stream.
- DO ECHO

  If a WILL ECHO is received, a DONT ECHO will be sent in response.
- DO ENVIRON

  Indicates a desire to be able to request environment variable information. (See RFC 1408.)
- DO LFLOW

  Requests that the client handle flow control characters remotely.
- DO LINEMODE

  Supports requests that the client do line-by-line processing.
- DO NAWS

  Requests that the client inform the server when the window size changes.
- DO NEW-ENVIRON

  Indicates a desire to be able to request environment variable information. (See RFC 1572.)
- DO SGA

  Indicates that it does not need to receive IAC GA, the go ahead command.
- DO TERMINAL-TYPE

  Indicates a desire to be able to request the name of the type of terminal that is attached to the client side of the connection.
- DO TERMINAL-SPEED

  Indicates a desire to be able to request information about the speed of the serial line to which the client is attached.
- DO TIMING-MARK

  Only supported if the client responded with WONT LINEMODE. If the client responds with WILL TM, then it is assumed that the client will support kludge linemode. It is not used for any other purposes.

- DO XDISPLOC

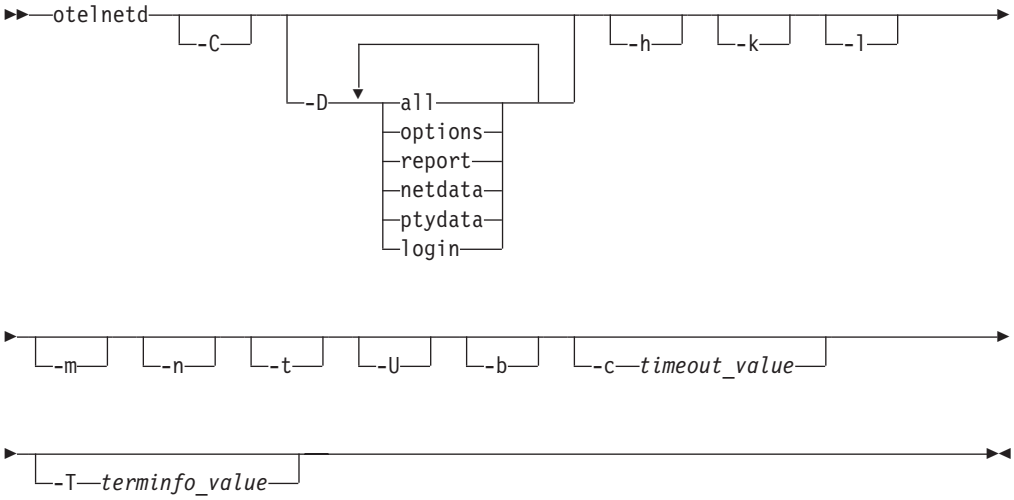
  Indicates a desire to be able to request the name of the X Window System display that is associated with the Telnet client.

otelnetd

Note: The user ID associated with the daemon in /etc/inetd.conf requires superuser authority. Refer to *z/OS UNIX System Services Planning* for a description of the types of authority defined for daemons.

The following syntax is used in the /etc/inetd.conf file to define the arguments used to invoke otelnetd.

Syntax



Parameters

-C Prints user messages in uppercase. There are several exceptions. Messages issued are start-up are not affected by the -C option because the -C option is not processed during the start-up. Also, data transmittal messages will not be uppercase. Data transmittal messages are generated from the -D netdata option or the -D ptydata option.

-D The following suboptions apply to -D:

options
Prints information about the negotiation of Telnet options. This information is used for debugging purposes. This suboption allows telnetd to generate debugging information to the connection, which allows the user to view telnetd activity.

report Prints the options information and additional information about processing. This information also includes print information designated for suboption=options. This can be used for debugging purposes. This suboption telnetd to generate debugging information to the connection, which enables the user to view telnetd activity.

netdata
Displays the data stream received by telnetd. This information is used for debugging purposes. It allows telnetd to generate debugging information to the connection, which enables the user to view telnetd activity.

ptydata
Displays the data stream written to the pty. This information is used for debugging purposes. It allows telnetd to generate debugging information to the connection, which enables the user to view telnetd activity.

all Enables options, report, netdata, and ptydata.

login Records login and logout activity to syslogd facility auth using message EZYTU36I.

-h Disables the display of the /etc/banner file at the user's terminal.

-k

Disables kludge linemode. The server normally attempts to use kludge linemode when the -l option was specified, but the client does not support line mode. Use the -k option when there are remote clients that do not support kludge linemode, but pass the heuristic for kludge line mode support (for example, if they respond with WILL TIMING-MARK in response to a DO TIMING-MARK). This option does not disable kludge line mode when the client requests it. This is accomplished by the client sending DONT SUPPRESS-GO-AHEAD and DONT ECHO.

-l Specifies linemode, which tries to force clients to use linemode. If the LINEMODE option is not supported and the -k option was not specified, it will attempt to use kludge linemode.

Notes:

1. Many clients decline the server's request to operate in linemode.
2. Linemode is not appropriate for full-screen applications like the z/OS UNIX vi editor.

-m Enables the creation of a forked or spawned process to coexist in the same address space. This option can improve performance because the user's login shell runs in the same address space as otelnetd.

-n Disables TCP keep-alives. Normally, telnetd enables the TCP keep-alive mechanism to probe connections that have been idle for some time to determine if the client is still there. In this way, idle connections from machines that have crashed or can no longer be reached can be cleaned up. The cleanup of disabled connections is controlled by the presence of the KEEPALIVEOPTIONS statement in the TCPIP profile.

-t Specifies internal tracing. It also activates the REPORT option, as if the user also specified -D Report.

-U Causes telnetd to drop connections from any IP address that cannot be mapped back into a symbolic name via the gethostbyaddr(3) routine.

-b Forces the server to DO BINARY in the first pass during negotiations with the client.

-c *timeout_value*
Specifies the number of seconds to wait before terminating the Telnet session for inactive connections. The *timeout_value* is a value between 1 and 86400 seconds.

-T *terminfo_value*
Sets the TERMINFO environment variable to the specified values at startup. This option is needed when terminfo definitions are located in nonstandard directories.

SMF Record Handling

The SMF records generated are the typical set of records that MVS generates for start of job (login) and end of job (logoff). Additionally, interval records might be issued during the life of the user login. These records are SMF TYPE 30 and TYPE 72 and not the TYPE 118 in the current Telnet server. The process of issuing these records is external to the specific daemons.

BPX.DAEMON Considerations

If the BPX.DAEMON facility class is defined, perform the following additional configuration steps:

1. Provide read access to BPX.DAEMON for the user ID specified in /etc/inetd.conf for otelnetd.
2. Define hlq.SEZALINK to program control.
3. Define the C run-time library, hlq.SCEERUN to program control.

Refer to *z/OS UNIX System Services Planning* for more information about the BPX.DAEMON facility class, the security product commands used to perform the required configuration, and the diagnosis procedure for resolving related problems.

Chapter 7. Transferring Files

FTP

The File Transfer Protocol (FTP) allows a user to copy files from one machine to another. The protocol allows for data transfer between the client (the end user) and the server in either direction. In addition to copying files, the client can issue FTP commands to the server to manipulate the underlying file system of the server (for example, to create or delete directories, delete files, rename existing files, and so on.) FTP is the most frequently used TCP/IP application for moving files between computers.

Copying files from one machine to another is one of the most frequently used operations. The data transfer between client and server can be in either direction. The client can send a file to the server machine. It can also request a file from this server.

To access remote files, the user must identify himself or herself to the server. At this point the server is responsible for authenticating the client before it allows the file transfer.

From an FTP user's point of view, the link is connection-oriented. FTP uses TCP as a transport protocol to provide reliable end-to-end connections. For example, it is necessary to have both hosts running TCP/IP to establish a file transfer.

The FTP server listens for connection requests on the well-known port 21, with data transfers occurring on the well-known port 20. Port 21 is also used to communicate with the server (for example, to issue commands, login, and initiate transfers). Port 20 is used for the data connection. The data connection is used to transfer data between the client and the server. When logging into the remote host, the user must have a user name and a password to access files and directories. The user who initiates the connection assumes the client function, while the server function is provided by the remote host.

Note: When FTPD accepts a connection on port 21, the server will fork() a new address space to handle the request. This address space assumes the name of the user. For example, if a user logs into an FTP server with the user ID of TCP0001, then the FTP server address servicing the request is also known as TCP0001. You can use the environment variable _BPX_JOBNAME to change this behavior. If coded, all forked FTP address spaces are known as the job name specified by the _BPX_JOBNAME variable. Having a common naming convention for your installation's FTP address spaces may be needed if your installation uses syslogd isolation or has other workload management requirements.

Configuring PROFILE.TCPIP for FTP

The FTP server can be started automatically when the TCP/IP address space is started by specifying the name of the FTP server catalogued procedure in the AUTOLOG statement. In the following example, if your procedure is called FTPD the following statement allows TCP/IP to issue the MVS start command for procedure FTPD. The job name of FTPD1 will be used on the port statement shown below. If the daemon job name is less than eight characters, the FTP daemon forks() a process that has the job name of the original daemon appended with the number ″1″.

```
AUTOLOG
  FTPD JOBNAME FTPD1
ENDAUTOLOG
```

To reserve ports 21 and 20 for the FTP server, add the following:

```
 PORT
   21 TCP FTPD1             ; FTP server control port
   20 TCP OMVS NOAUTOLOG    ; FTP server data port
```

In addition, because FTPD1 is on the PORT and AUTOLOG statements, if FTP ends, TCP/IP will restart the FTPD.

To allow the FTP data connections to time out when there has been no activity on the data connection for a certain amount of time, set the KEEPALIVEOPTIONS statement in the PROFILE.TCPIP data set to the connection timeout value desired:

```
KEEPALIVEOPTIONS INTERVAL number_of_minutes ENDKEEPALIVEOPTIONS
```

Be careful when choosing a timeout interval for the KEEPALIVEOPTIONS statement because this value will affect *all TCP* connections at this host for which KEEPALIVEOPTIONS has been activated, not just the FTP data connections.

For more information about the AUTOLOG, PORT, and KEEPALIVEOPTIONS statements, refer to *z/OS Communications Server: IP Configuration Reference*.

Configuring ETC.SERVICES

The ETC.SERVICES data set contains the relationship between service names (servers) and port numbers in the z/OS UNIX environment. If necessary, update your ETC.SERVICES file to include the control port that the FTP server is to use. See “Chapter 1. Configuration Overview” on page 3 for search order used to locate the ETC.SERVICES file. For example, add the following:

```
ftp 21/tcp
```

Notes:

1. In the ETC.SERVICES file, only one port (the one for the control connection) is listed.
2. The port specified for FTP in the ETC.SERVICES file can be overridden by the FTP start parameter PORT nnnn and should match the PORT statement in the PROFILE.TCPIP.
3. If the ETC.SERVICES file is changed such that a port other than 21 is specified, then that port will become the FTP port for that z/OS host.

Configuring /etc/syslog.conf

Note: For ftpd syslog, you should consider the fact that ftpd writes log messages to the system console if syslogd is not running. If you enable ftp server traces without syslogd active, large amounts of data might be written to the system console.

The daemon.*priority* entries in /etc/syslog.conf determine where FTP messages and trace entries are written. The FTP server issues info, warning, and error messages. All trace entries are written with debug priority. To direct trace entries (and all messages) to /tmp/daemon.trace, include the following in /etc/syslog.conf:

```
*.*.daemon.debug   /tmp/daemon.trace
```

Log messages can be isolated within syslogd. For FTP, an installation might want FTP log messages to be written to different files depending on the user ID, or separately for the FTP daemon. If FTP messages are to be isolated for user1, use the first statement below. If FTP messages are to be logged for all the FTP applications, use the second statement below.

```
user1.*.daemon.debug /tmp/daemon.trace
*.FTPD*.daemon.debug /tmp/daemon.trace
```

In the above statement, it is assumed that _BPX_JOBNAME is set to FTPD.

Configuring the FTPD Cataloged Procedure

Update the FTP cataloged procedure FTPD by copying the sample in *hlq*.SEZAINST(FTPD) to your system or recognized PROCLIB and modifying it to suit your local configuration. The EXEC PARM=, SYSFTPD DD, and SYSTCPD statements must be updated.

See “Configuring FTP.DATA” on page 307 to configure SYSFTPD DD and “Configuring TCPIP.DATA for FTP” on page 307 to configure SYSTCPD DD.

The system parameters required by the FTP server are passed by the PARM parameter on the EXEC statement of the FTPD cataloged procedure. Add your parameters to `PARMS='` in the PROC statement of the FTPD cataloged procedure, making certain that each parameter is separated by a blank and all parameters are in uppercase

For example: `//FTPD PROC MODULE='FTPD',PARMS='TRACE ANONYMOUS PORT 21'` tells FTP to start up with TRACE active, ANONYMOUS support enabled, and use control PORT 21.

Security Considerations for the FTP Server

Consider the following for security:

- User IDs

  To log into the FTP server, a user ID must have an z/OS UNIX UID.
- MVS Network Access Controls
  - If PortAccess or NetAccess is used to SAF resource secure TCP ports or networks, see the NETACCESS statement in *z/OS Communications Server: IP Configuration Reference* for more information.
- The FTPD cataloged procedure must be:
  - Defined to the security program.
  - Added to the RACF started class facility or the started procedures table. The user ID associated with the FTP server started class must have a UID of 0.
  - See SEZAINST(EZARACF) for more information on SAF resource requirements needed for FTP.
- Terminal Access

  The terminal ID passed from FTP to RACF is an 8-byte hexadecimal character string containing an IP address. RACF interprets this as a terminal logon address and rejects it if it is not previously defined. For example, the IP address 163.97.227.17 is translated to X'A361E311'.

  Therefore, if the SETROPTS TERMINAL(NONE) setting is used in RACF, you must define profiles for the IP addresses in class TERMINAL to avoid problems

when trying to FTP to MVS. You must translate all the IP addresses of any clients connecting to FTP servers to hexadecimal character strings and add them to the class TERMINAL.

To allow access by all addresses starting with “163,” define a profile for all addresses in the 163.97.227 subnet:

```
RDEFINE TERMINAL A361E3* UACC(READ)
```

If your RACF SETROPTS options are TERMINAL(READ), all terminals are allowed access to your system, and you do not have to add extra resource definitions to your RACF data base.

For more information, see *z/OS UNIX System Services Planning* and the *z/OS SecureWay Security Server RACF Security Administrator's Guide*.

Defining Environment Variables for the FTP Server (Optional)

The FTP server optionally uses environment variables to identify the translate table data sets to be used for the control and data connections. These environment variables are used to override a default naming convention as described below.

_FTPXLATE_name Used for Translation

In your FTP.DATA file, you can use the CCXLATE or XLATE statements to specify a name that corresponds to a particular data set that is to be used for the initial translate tables for the control or data connections.

FTP will look for an environment variable defined as *_FTPXLATE_name=fully_qualified_dsn*, where *name* must be one to eight uppercase characters or numbers, and *fully_qualified_dsn* can be a fully-qualified MVS data set name or HFS file name.

If the environment variable exists, FTP will use the data set name defined by the environment variable. If no such environment variable is defined, FTP will use the data set name *hlq.name*.TCPXLBIN.

Similarly, from any client you can issue SITE XLATE= to set the translate tables for the data connection for that particular FTP session. The FTP server will look for an environment variable called *_FTPXLATE_name*. If the environment variable does not exist, the server will look for a data set called *hlq.name*.TCPXLBIN.

Note: The CCXLATE and XLATE statements and SITE XLATE command are not case-sensitive, but the name of the optional environment variable is case-sensitive and must be in uppercase or FTP will not recognize it.

TZ and other UNIX Environment Variables

You can use the ENVAR runtime option in your FTPD start procedure to set environment variables for the FTP server. For information on using the ENVAR runtime option to set environment variables, see *z/OS C/C++ Programming Guide*. The following example shows how to specify environment variables in your FTPD started procedure:

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR('"TZ=EST")/&PARMS')
```

_BPX_JOBNAME

An installation that wants all FTP forked tasks to have similar job names needs to set the _BPX_JOBNAME environment variable. WorkLoad Manager (WLM),

accounting, and isolation of syslogd messages as reasons an installation might not want to have each FTP logged-in user to have a job name of its user ID.

The following example sets all FTP forked() tasks to have the job name of FTPD followed by a suffix of 1, 2, 3, 4, and so on.

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("_BPX_JOBNAME=FTPD"',
//           '"TZ=EST")/&PARMS')
```

_BPXK_SETIBMOPT_TRANSPORT for Affinity to a Specific Stack

As discussed in "Generic Server Versus Server with Affinity for a Specific Transport Provider" on page 39, if an installation wants to ensure that FTP has an affinity to a TCP/IP stack, the _BPXK_SETIBMOPT_TRANSPORT keyword should be used.

The example below sets the FTP server to have an affinity to TCPIEOE.

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE"',
//           '"TZ=EST")/&PARMS')
```

Configuring TCPIP.DATA for FTP

The following five statements are used by the FTP server:

DATASETPREFIX
Specifies HLQ for dynamic allocation

DOMAINORIGIN
Specifies the domain name to be appended to host name

HOSTNAME
Specifies the TCP host name

LOADDBCSTABLES
Specifies the DBCS tables used by the client and server

MESSAGECASE
Specifies the case that messages should be displayed in

See "Chapter 1. Configuration Overview" on page 3 for information about TCPIP.DATA or refer to *z/OS Communications Server: IP Configuration Reference* for information about these statements.

Configuring FTP.DATA

The FTP.DATA data set is optional. The FTP daemon looks for this data set during initialization, following this sequence:

1. A data set specified by the //SYSFTPD DD statement
2. *ftpserve_job_name*.FTP.DATA
3. /etc/ftp.data
4. SYS1.TCPPARMS(FTPDATA)
5. *hlq*.FTP.DATA data set

It is not necessary to include all statements in the FTP.DATA data set. Only include the statements if the default value is not what you want, because the default will be used for any statement not included in the FTP.DATA data set.

Several of the FTP server parameters can be changed during an FTP session by the client issuing the SITE subcommand. See *z/OS Communications Server: IP User's Guide* for more information. The FTP client has an FTP.DATA data set which can also be used to change the defaults for the FTP client local site parameters. See the *z/OS Communications Server: IP User's Guide* for more information about using the FTP.DATA data set for the FTP client local site parameters.

Data Set Attributes

Data set attributes play a significant role in FTP performance. If your environment permits, tune both BLOCKSIZE and LRECL according to the following recommendations:

- Use half a track as the block size.
- For IBM 3380 DASD, use 23424 as the block size with an LRECL of 64 bytes.
- For IBM 3390 DASD or IBM9334, use 27968 as the block size with an LRECL of 64 bytes.
- Use FB as the data set allocation format.
- Use cached DASD controllers.
- If your environment permits, use a pre-allocated data set for FTP transfers into MVS.

The following configuration data statements apply to FTP server's allocation of data sets.

- AUTOMOUNT
- BLKSIZE
- BUFNO
- CONDDISP
- DATACLASS
- DCBDSN
- DIRECTORY
- LRECL
- MGMTCLASS
- MIGRATEVOL
- PRIMARY
- RECFM
- RETPD
- SECONDARY
- SPACETYPE
- STORCLASS
- UCOUNT
- UMASK
- UNITNAME
- VCOUNT
- VOLUME

Refer to *z/OS Communications Server: IP Configuration Reference* for more detailed information about these keywords. Also, see the example below for a short explanation of the statements.

Some of these allocation variables might provide duplicate information. FTP passes all variables that are specified to z/OS's dynamic allocation function and lets it determine which of the specifications take precedence. The only exceptions to this are the following:

- If the data set organization is physical sequential, then directory blocks are not sent.
- If neither primary nor secondary space quantities are specified, then the allocation units value is not sent.

For example, the model DCB (DCBDSN) might have a record format (RECFM) that differs from the record format specified by a data class and from the one explicitly specified by the client. The order of precedence for dynamic allocation variables are as follows:

1. Any FTP.DATA statements or SITE parameters explicitly specified or defaulted.
2. Any attributes picked up from the model DCB and not otherwise explicitly specified.
3. Any attributes picked up from the data class and not previously derived from 1 and 2 above.
4. Any system allocation defaults.

Specifying Attributes for New MVS Data Sets

When allocating new data sets, there are two methods you can use to specify the data set attributes. You can individually use the data set attribute parameters with the SITE command or the statements in the FTP.DATA data set. Or, if your system programmer has used the Storage Management System to group together default attributes into named classes, you can specify those class names on the DATACLASS, STORCLASS, and MGMTCLASS statements.

Dynamic Allocation

The FTP server allows a client program to dynamically allocate a new physical sequential data set or a partitioned data set (PDS) for the purpose of transferring data to be written to that data set. The following optional allocation variables can be used to override and turn off the hard-coded defaults that affect the allocation of the data set.

Variable	**FTP.DATA statement**
allocation units	SPACETYPE
blocksize	BLKSIZE
data class	DATACLASS
directory blocks	DIRECTORY
logical record length	LRECL
management class	MGMTCLASS
model DCB values	DCBDSN
primary space	PRIMARY
secondary space	SECONDARY
unit count	UCOUNT
volume count	VCOUNT
record format	RECFM
retention period	RETPD
storage class	STORCLASS
unit	UNITNAME
volume serial number or list	VOLUME

Some of these allocation variables might provide duplicate information. For example, the model DCB might have a record format (RECFM) that differs from the

record format specified by a data class and from the one explicitly specified by the client. FTP passes all variables that are specified to dynamic allocation and lets it determine which of the specifications take precedence. The following list describes the exceptions to that policy:

- If neither the primary nor secondary space quantity is specified, then the allocation units value is not sent.
- If the data set organization is physical sequential, then directory blocks specification is not sent.
- Otherwise, all variables are sent to dynamic allocation where the order of precedence is:
  1. Any FTP.DATA statements or SITE parameters explicitly specified or defaulted
  2. Any attributes picked up from the model DCB and not otherwise explicitly specified
  3. Any attributes picked up from the data class and not previously derived from 1 or 2
  4. Any allocation defaults

Storage Management Subsystem (SMS)

You can specify one or more of the following SMS classes to manage characteristics that are associated with or assigned to data sets.

- Data class is an SMS construct that an installation can define to control data set allocation attributes used by SMS for the creation of data sets. An installation can override all or part of an SMS DATA CLASS definition by using FTP.DATA statements. Note that there is an order of precedence for dynamic allocation. (See “Data Set Attributes” on page 308 for more information on the precedence.) The fields listed are available attributes that serve as a template for allocation. Each is ***optional*** and is overridden by any explicit specification of FTP allocation variables or by a model DCB (DCBDSN).

Variable	FTP.DATA statement
directory blocks	DIRECTORY
logical record length	LRECL
primary space	PRIMARY
record format	RECFM
retention period	RETPD
secondary space	SECONDARY

  Note: If either primary or secondary space is explicitly specified, then the primary and secondary values from data class are not used.
- Management class (MGMTCLASS) is an SMS construct that determines DFHSM action for data set retention, migration, backup, and release of allocated but unused space. Management class replaces and expands attributes that otherwise would be specified. That is, management class might override any other specification of retention period.
- Storage class (STORCLASS)is a list of storage performance and availability services requests for an SMS-managed data set that SMS attempts to honor when selecting a volume or volumes for the data set. It might conflict with an explicit specification of volume and unit. If storage class is used, then volume and unit should be unspecified.

Translation of Data

Translation of data from ASCII to EBCDIC, for print control characters, etc are important to ensure that data read from or written to the z/OS system are in the

correct format. The following statements apply to translation of data for the FTP server. Refer to *z/OS Communications Server: IP Configuration Reference* for more information on these statements. The statements are:

- ASATRANS
- CCXLATE
- CTRLCONN
- SBDATACONN
- UCSHOSTCS
- UCSSUB
- UCSTRUNCT
- XLATE

Accounting

The following parameters apply to SMF data:

- SMF
- SMFAPPE
- SMFDEL
- SMFEXIT
- SMFJES
- SMFLOGN
- SMFREN
- SMFRETR
- SMFSQL
- SMFSTOR

Refer to *z/OS Communications Server: IP Configuration Reference* or the example for more information on these statements.

Configure the FTP Server for SMF (Optional)

The FTP server can write type 118 (X'76') SMF records to record transactions made by the FTP server. SMF records are independent of the IP connection. They are created for host connections. SMF records can be written for the following commands:

- APPEND
- DELETE
- RENAME
- RETRIEVE
- STORE
- STORE UNIQUE

Information about the previous commands can be recorded for:

- FTP server running in normal data transfer mode (FILETYPE=SEQ)
- FTP server running remote job submission (FILETYPE=JES)
- FTP server running Structured Query Language (SQL) queries (FILETYPE=SQL)
- Any combination of SEQ, JES, and SQL

For commands involving data transfer (APPEND, GET, PUT, RETR, or STOR) an SMF record will be written for both successfully and unsuccessfully completed data transfer commands which have begun data transfer. For data transfer commands

which have completed unsuccessfully, the byte count of transmission field (offset 68) will contain the number of bytes transferred before the failure, and the recent server reply field (offset 73) will contain the 3-digit error reply code sent to the client.

The FTP server can also write SMF records when a logon attempt fails.

The capability also exists for a user-written exit routine to get control before the SMF records are written.

If you want the FTP server to write type 118 (X'76') SMF records, you must include at least one of the SMF subtype statements (SMF, SMFAPPE, SMFDEL, SMFLOGN, SMFREN, SMFRETR, or SMFSTOR) in the FTP.DATA data set.

If SMF subtype statements are not coded in the FTP.DATA data set, then no SMF records are written by the FTP server.

DB2 and JES

The following statements are used when FTP interfaces with DB2® and JES, respectively. For more information, refer to *z/OS Communications Server: IP Configuration Reference* and the optional steps in this chapter.

- DB2
- DB2PLAN
- SPREAD and SQLCOL
- JESLRECL
- JESPUTGETTO
- JESRECFM
- JESINTERFACELEVEL

Transferring Data

CHKPTINT, DEST, FILETYPE, and WRAPRECORD are used for determining how data is transferred. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

DEST specifies the NJE destination for a PUT command. INACTIVE is used to set the value of the inactivity time or to disable inactivity timeouts. TRACE is used for tracing the FTP server. WLMCLUSTERNAME is used register the FTP daemon with the DNS/WLM sysplex.

Configuring the Optional FTP User Exits

The FTPSMFEX User Exit

The FTP server SMF user exit is called before an SMF record that contains information about an FTP server session is written to the SYS1.MANx data set. The user exit allows site specific modifications to the record and controls whether the record is written to the SYS1.MANx data set.

The FTCHKIP User Exit

The FTCHKIP user exit is called when a user attempts to log in to the FTP server or when a user issues the OPEN command to establish a new connection. The IP

address and PORT number of both the local and remote host is passed to the user exit. An installation can use this exit to determine if a particular IP address or port number is allowed to access the FTP site. The message `421 User Exit rejects open for connection<` is sent to the user if the connection is denied.

The FTCHKPWD User Exit

The FTCHKPWD user exit is called immediately after the user enters the password or e-mail address during login to the FTP server. The following information is passed to the exit:

- The user ID
- The user password or an asterisk (*) if an e-mail address is entered instead of a password
- A userdata buffer

  If an e-mail address is entered to log in, the userdata buffer contains the e-mail address.

The exit can be used to restrict access to a site based on user ID and password. The message `530 User Exit rejects logon by 'xxxxx'` is sent to the user if the logon is denied.

The FTCHKCMD User Exit

The FTCHKCMD user exit is called whenever the user enters an FTP command. The following information is passed to the user exit. (Prior to CS for z/OS V2R10, only three parameters were passed to FTPCHKCMD.)

- The user ID
- The FTP command to be issued
- The command's arguments
- The directory type (MVS or HFS)
- The current working directory
- The FILETYPE (SEQ, JES, or SQL)
- A buffer

The user exit allows an installation to modify the arguments of an FTP command or to deny a user from issuing the command. For example, if a user issues a DIR * ftp command, the exit can either deny the command or modify it to DIR 'USER1.*'. The message `500 User Exit denies Userid xxxxx from using Command yyy` is sent to the user if the command is denied.

The FTCHKJES User Exit

FTCHKJES is called if the server is in FILETYPE=JES mode and the client tries to submit a job. The user ID and the job being submitted are passed to the exit. The exit can allow or refuse the job to be submitted to the JES internal reader. For example, the exit can look for a USER= parameter on the JOB statement and check it against the client's user ID. The message `550 User Exit refuses this job to be submitted by userid` is sent to the user if the remote job submission is denied.

The FTPOSTPR User Exit

FTPOSTPR is called after execution of the FTP commands RETR, STOR, STOU, APPE, DELE, and RNTO. The user exit is passed the user ID, the client IP address, the client port number, the current directory type, the length of the parameter string, the current working directory, the current file type, the FTP reply

code, a buffer containing the FTP reply string, the FTP command code, the current CONDDISP setting and the close reason code. The exit allows for post processing of user written batch jobs.

Customizing the FTP-to-JES Interface (Optional)

Before customizing the FTP-to-JES interface, complete JES customization. For example, JESJOBS is a Security Access Facility (SAF) class that controls which users can submit jobs to JES. JESSPOOL is the SAF that controls which users can access output jobs. Customize these SAF classes before beginning customization of the FTP-to-JES interface.

If FTP.DATA does not change the JESINTERFACELevel to 2, the FTP server will behave as in previous releases. FTP clients will be allowed to submit jobs to JES, retrieve held output that matches their logged in user ID plus one character and delete held jobs that match their logged in user ID plus one character. If JESINTERFACELevel is set to 2, then FTP clients have the ability to retrieve and delete any job in the system permitted by the security access facility (SAF) resource class JESSPOOL. For that reason, JESINTERFACELevel=2 should only be specified if the proper JES and SDSF security measures are in place to protect access to JES output. The SAF controls used for JESINTERFACELevel=2 are essentially a subset of those used by SDSF. Therefore, if an installation has customized SAF facilities for SDSF, then they are configured for FTP JES level 2.

JESSPOOL defines resource names as <nodeid>.<userid>.<jobname>.<Dsid>. <dsname>. An FTP client can delete an output job if it has ALTER access to the resource that matches its nodeid, userid, and job name. If the FTP client has READ access to the resource, it can retrieve or GET the job output. No JESSPOOL access is required to DIR, LIST, or NLST jobs. In summary, JESSPOOL access is the normal SAF resource class that allows access to job output. Refer to *OS/390 V2R7.0 JES2 Initialization and Tuning Guide* for more information on JES security.

To control access to DIR, LIST, or NLST of jobs, the SDSF resources are used. For example, JESSTATUS can be changed by an FTP client via the SITE command to filter jobs in INPUT, ACTIVE, or OUTPUT state. The SDSF resources checked for these states are ISFCMD.DSP.INPUT.jesx, ISFCMD.DSP.ACTIVE.jesx, and ISFCMD.DSP.OUTPUT.jesx, respectively. At login time (USER command), the default value is set to ALL if READ access is allowed to all three classes. Otherwise it attempts to set it to OUTPUT, ACTIVE, and then INPUT if the appropriate READ access is allowed. If no READ access is allowed to any of the classes, JESSTATUS is set to OUTPUT but JESOWNER and JESJOBNAME cannot be changed from the default. In this way, SAF controls can be put in place to limit FTP clients to whatever status of jobs an installation requires.

There are two additional access filters used by the FTP server to control access to jobs. At login time JESOWNER will have the value of the logged in user ID. JESJOBNAME will have the value of the logged in user ID plus an asterisk (*). JESOWNER and JESJOBNAME are authorized via READ access to RACF profiles ISFCMD.FILTER.OWNER and ISFCMD.FILTER.PREFIX respectively. An FTP client who has READ access to ISFCMD.FILTER.OWNER will be allowed to change the JESOWNER parameter via the SITE command. An FTP client who has READ access to ISFCMD.FILTER.PREFIX will be allowed to change the JESJOBNAME parameter via the SITE command.

For example, to allow all users except USER1 to be allowed to change JESOWNER enter the following:

```
SETROPTS CLASSACT(SDSF) REFRESH
RDEFINE SDSF (ISFCMD.FILTER.OWNER) UACC(READ)
PERMIT ISFCMD.FILTER.OWNER ACCESS(NONE) CLASS(SDSF) ID(USER1)
SETROPTS CLASSACT(SDSF) REFRESH
```

Also, because SDSF and JESSPOOL class resource names are used to determine access to setting search filters and to data (respectively), users can be allowed to use the DIR command to display jobs but not allowed to GET jobs. In situations like this, a DIR command will show job status information but not the number of spool files, return code, and so on. For example:

```
dir j65
>>> PORT 127,0,0,1,4,10
200 Port request OK.
>>> LIST j65
125 List started OK for JESJOBNAME=USER1*, JESSTATUS=ALL and JESOWNER=USER1
JOBNAME  JOBID    OWNER    STATUS CLASS
USER1W   JOB00065 USER1    OUTPUT A        ABEND=0C1 spool files not accessible
250 List completed successfully.
Command:
```

For more information on SDSF security, refer to *z/OS SDSF Operation and Customization*.

Configuring the FTP Server for Anonymous Logins (Optional)

You can configure the FTP server to accept anonymous logins. A login is anonymous when the remote user specifies USER ANONYMOUS instead of an FTP user ID. To enable anonymous logins, add the ANONYMOUS statement to the server FTP.DATA data set.

You can specify three levels of anonymous support via the ANONYMOUSLEVEL keyword. ANONYMOUSLEVEL 1 is the default, and is equivalent to anonymous login support provided by releases prior to OS/390 V2R10. That is, the ANONYMOUS statement is supported. If no operands are specified on the ANONYMOUS statement, the anonymous user needs no password and has unrestricted access to the MVS and HFS file systems.

You can specify ANONYMOUSLEVEL 2, but this is not recommended. ANONYMOUSLEVEL 2 is provided for migration purposes only. Consider ANONYMOUSLEVEL 3 if ANONYMOUSLEVEL 1 does not meet your anonymous login security requirements.

If you specify ANONYMOUSLEVEL 3, the anonymous user cannot issue the USER command to leave anonymous mode, nor can another user issue USER anonymous to enter anonymous login mode. If you specify ANONYMOUSLEVEL 3 and STARTDIRECTORY HFS in FTP.DATA, the anonymous user's HFS access is restricted to the anonymous user's home directory and home directory subtrees.

The ANONYMOUSLEVEL 3 server recognizes additional keywords that restrict the anonymous user's access to FTP resources. These keywords are ignored when ANONYMOUSLEVEL is less than three:

- ANONYMOUSFILEACCESS allows the system programmer to preclude access to either the HFS or MVS file systems.

- ANONYMOUSFILETYPEJES, ANONYMOUSFILETYPESQL, and ANONYMOUSFILETYPESEQ control whether the anonymous user can set filetype JES, SQL, or SEQ, respectively.
- ANONYMOUSHFSFILEMODE defines the mode bits used for files written to the HFS.
- ANONYMOUSHFSDIRMODE defines the mode bits used for directories created in the HFS.

Finally, when ANONYMOUSLEVEL is set to three, the user's e-mail address is requested in lieu of a password when:

- ANONYMOUS is specified without any parameters.
- ANONYMOUS is specified with user ID/password.

Control the degree of verification of the e-mail address an anonymous user enters as password by using the EMAILADDRCHECK keyword in FTP.DATA. Refer to *z/OS Communications Server: IP Configuration Reference* for details about the EMAILADDRCHECK keyword. The e-mail address entered is logged to the syslog daemon and is also passed to a user exit routine, FTCHKPWD, for user processing.

If you choose ANONYMOUSLEVEL greater than one and you choose STARTDIRECTORY HFS, you must create an anonymous directory structure in the HFS.

Creating an Anonymous Directory Structure in the HFS

The sample shell script, ftpandir.scp, will create an anonymous directory structure for you, containing required and optional structures. Or, a superuser can create the anonymous directory structure. In this section, the steps a superuser would follow to create an anonymous HFS directory structure are outlined.

For the following steps, assume that the RACF user ID that is used when an anonymous user logs in is called GUEST, that the HOME directory in that user's OMVS segment in RACF is /u/guest, and that FTP.DATA contains a statement similar to this: ANONYMOUS GUEST

1. Create a bin subdirectory in the anonymous root containing the executables ls and sh. This is a required directory. ls can be copied from the standard directory. sh is part of the standard MVS search order, so you need only create an empty file with the sticky bit.

   The following example shows how to create ls and sh in the user GUEST's home directory:

   ```
   ===> cd /u/guest
   ===> mkdir bin
   ===> chmod 711 bin
   ===> cd bin

   ===> cp /bin/ls ls
   ===> chmod 711 ls
   ===> touch sh
   ===> chmod 711 sh
   ===> chmod +t sh
   ```

   An ls -al command should give the following results. Owner and group attributes may be different in your system.

   ```
   # ls -al
   total 280
   drwx--x--x   2 USER22   0  8192 Sep 21 17:39 .
   ```

```
drwx--x--x   7 USER22   0  8192 Nov  1 14:44 ..
-rwx--x--x   1 USER22   0  126976 Sep 21 17:39 ls
-rwx--x--t   1 USER22   0  0 Sep 21 17:39 sh
```

2. Create a usr/sbin sudirectory of the anonymous root containing the executable file ftpdns. This is a required subdirectory. The file ftpdns can be empty with the sticky bin on.

   The following example is for anonymous user GUEST:

   ```
   ===> cd /u/guest
   ===> mkdir usr
   ===> chmod 711 usr

   ===> cd usr
   ===> mkdir sbin
   ===> chmod 711 sbin
   ===> cd sbin
   ===> touch ftpdns
   ===> chmod 711 ftpdns
   ===> chmod +t ftpdns
   ```

   If you do not configure the subdirectories, bin and usr/sbin, and their contents correctly, the FTP server will not be able to accept anonymous logins and message EZYFT731 will be displayed.

3. Set up the public directory structure. This is a required directory.

   This is the directory structure into which you place files that can be downloaded by the anonymous FTP user. It does not have to be named pub; it can be any name you choose. A general convention for anonymous FTP sites is to call it pub:

   ```
    ===> cd /u/guest
   ===> mkdir pub
   ===> cd pub
   ```

   If you want to structure the files you allow to be accessed, you can create multiple subdirectories underneath this directory.

   For simplicity, assume a single level directory, the pub directory. Into this directory you copy the files you want to allow the anonymous user to download:

   ```
   ===> cp /x/y/z/prodinfo1.txt prodinfo1.txt
   ===> cp /x/y/z/prodinfo2.txt prodinfo2.txt
   ===> cd ..
   ```

   Make sure that the permission bits are set correctly by using the following shell command when are positioned in the /u/guest directory. This will set the permission bits of all files in the pub directory and its subdirectories to 755:

   ```
   ===> chmod -R 755 pub
   ```

   If your system does not require an incoming or extract directory, the system is configured for anonymous FTP. An ls —al command of the pub directory should give the following results:

   ```
   drwxr-xr-x   3 IBMUSER  SYS1  8192 May 13 21:15 .
   drwxr-xr-x   6 IBMUSER  SYS1  8192 May 20 14:51 ..
   -rwxr-xr-x   1 IBMUSER  SYS1  12 May 11 12:41 prodinfo1.txt
   -rwxr-xr-x   1 IBMUSER  SYS1  12 May 11 12:41 prodinfo2.txt
   ```

4. Set up an incoming directory (optional).

   If you want anonymous users to be able to upload files to your FTP server, you need some more setup. The objective is to allow an anonymous user to upload a file, but not to allow another anonymous user to download or even be aware of the existence of the file until after an administrative user has verified that the

content of the file is acceptable. You do not want your FTP server site to become a store-and-forward site for files of questionable ethical content.

Positioned at the /u/guest directory, a superuser issues the following shell command:

```
===> cd /u/guest
===> mkdir incoming
===> chmod 733 incoming
```

It does not have to be named incoming; it can be any name you choose. A general convention for anonymous FTP sites is to call it incoming.

The 733 permission bits means that a non-superuser cannot list the content of the incoming directory, but can write a file to it. Because the FTP server enforces a UMASK of 777 when an anonymous user logs in, these files will be written with permission bits 000, which means that they cannot be accessed by the anonymous user or by any other user except a superuser.

An FTP client user can normally change the UMASK via a SITE UMASK command or the user can change the permission bits of files he/she owns through a SITE CHMOD command.

If you define ANONYMOUSLEVEL 3, you can use the ANONYMOUSHFSDIRMODE keyword to set the permission bits of any directory created by an anonymous user, and the ANONYMOUSHFSFILEMODE to set the permission bits of any file created by an anonymous user.

If you do allow anonymous users to store files on your FTP server, you should ensure that the directory into which these files are stored is in an HFS that can fill up without impacting other work on your z/OS system. The best way to do that is to allocate the /u/guest/incoming directory in its own HFS data set. If an anonymous user uploads large amounts of data to the incoming directory, only this separate HFS will be filled up. Filling this separate HFS will prevent other anonymous users from storing new files on the server, but will not affect other functions on your system. At a minimum, you should make sure that the incoming directory is not in the same HFS as your /tmp directory.

5. Set up the extract directory (optional).

   If you need to make files available to certain anonymous users, but not to everyone, you can create a directory that cannot be listed, but files in it can be downloaded if the anonymous user knows the name of the file.

   Positioned at the /u/guest directory, a superuser issues the following shell commands:

   ```
   ===> cd /u/guest
   ===> mkdir extract
   ===> chmod 711 extract
   ```

   It does not have to be named extract; it can be any name you choose. A general convention for anonymous FTP sites is to call it extract.

   A superuser can then copy files into this directory, ensure they have permissions of 755, inform the intended anonymous user of the file name, and that user can then log on as anonymous and retrieve the file.

   An ls -al command at the /u/guest location should give the following result, if you created all four subdirectories:

```
drwxr-xr-x   6 IBMUSER  SYS1 8192 May 20 14:51 .
dr-xr-xr-x   6 IBMUSER  SYS1  0 Jun 10 15:43 ..
drwx--x--x   2 IBMUSER  SYS1  8192 May 11 12:44 bin
drwx--x--x   3 IBMUSER  SYS1  8192 May 11 13:39 extract
drwx-wx-wx   3 IBMUSER  SYS1  8192 May 25 09:35 incoming
drwxr-xr-x   3 IBMUSER  SYS1  8192 May 13 21:15 pub
```

Configure the Welcome Banner Page, Login, and Directory Message (Optional)

Starting in V2R10, the FTP server now provides support to allow FTP administrators to provide useful information about the site to FTP users. The following FTP.DATA statements are available:

- BANNER
- LOGINMSG
- ANONYMOUSLOGINMSG
- MVSINFO
- ANONYMOUSMVSINFO
- HFSINFO
- ANONYMOUSHFSINFO

You can use the LOGINMSG statement in FTP.DATA to point to a set of messages displayed when a known user logs in to FTP. Similarly, ANONYMOUSLOGINMSG can point to a set of messages displayed when an anonymous user logs in to FTP.

You can use the MVSINFO statement to point to a set of messages displayed when a known user changes the working directory to a particular MVS data set path. Likewise, use the ANONYMOUSMVSINFO statement to point to a set of messages displayed when an anonymous user changes working directory to a particular MVS data set path.

You can use the HFSINFO statement to point to a set of messages displayed when a client changes the working directory to a particular HFS directory. Likewise, use the ANONYMOUSHFSINFO statement to point to a set of messages displayed when an anonymous user changes working directory to a particular HFS directory.

Install the SQL Query Function (Optional) and Access the DB2 Modules

To use FTP to do SQL queries, bind the DBRM called EZAFTPMQ to the plan used by FTP, and grant execution privileges for that plan to PUBLIC. (The name of the plan can be specified by the DB2PLAN keyword in FTP.DATA or defaulted to EZAFTPMQ.) This FTP facility only performs SELECT operations on the DB2 tables. It does not perform UPDATE, INSERT, or DELETE.

Note: If secondary authorization for SQL queries is required, the DSN3SATH sample exit shipped by DB2 must be modified. The exit will return the primary AUTHID for requests originating from the FTP server.

The following sample job is provided in the FTOEBIND member of the SEZAINST data set. It can be used to enable the FTP server and client to do SQL queries.

```
//FTPSETUP JOB FTPSETUP,
//           CLASS=A,
//           NOTIFY=&SYSUID
```

```
//*************************************************************
//*
//*   File name:                tcpip.SEZAINST(FTOEBIND)
//*   SMP/E distribution name:   EZAFTPAB
//*
//*    Licensed Materials - Property of IBM
//*    This product contains "Restricted Materials of IBM"
//*    5647-A01 (C) Copyright IBM Corp.  1997.
//*    All rights reserved.
//*    US Government Users Restricted Rights -
//*    Use, duplication or disclosure restricted by GSA ADP Schedule
//*    Contract with IBM Corp.
//*    See IBM Copyright Instructions.
//*
//*  This JCL binds the EZAFTPMQ DBRM to the specified
//*  DB2 subsystem and allows execution of the
//*  EZAFTPMQ plan by PUBLIC.
//*
//*  The MVS OE FTP server and client use this plan. (See
//*  Usage note #7)
//*
//*
//*  Usage notes:
//*
//*    1.  You must execute this job from a user ID that has
//*        the authority to bind the EZAFTPMQ plan.
//*
//*    2.  Change the STEPLIB DD statement in the FTPBIND and
//*        FTPGRANT steps to reflect the DB2 DSNLOAD data set.
//*
//*    3.  Change the DB2 subsystem name in the FTPBIND and
//*        FTPGRANT steps from SYSTEM(xxx) to the
//*        installation defined DB2 subsystem name.
//*
//*    4.  Change the library parameter in the FTPBIND step from
//*        TCPIP.SEZADBRM to the installation defined TCPIP
//*        SEZADBRM library.
//*
//*    5.  Change the plan name in the FTPGRANT step from
//*        DSNTIAYY to reflect the plan associated with the
//*        program DSNTIAD.
//*
//*    6.  Change the library parameter in the FTPGRANT step
//*        from xxxxxx.RUNLIB.LOAD to reflect the library
//*        where the DSNTIAD program resides.
//*
//*    7.  You can bind the DBRM to a plan name other than EZAFTPMQ
//*        by changing the plan specified in the FTPBIND and
//*        FTPGRANT steps.  If you do this, you must use the
//*        DB2PLAN keyword in FTP.DATA to change the plan name
//*        used by the FTP server and/or client to the plan name
//*        specified here.
//*
//*************************************************************
//FTPBIND  EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN=xxxxxx.DSNLOAD,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSTSIN  DD   *
 DSN  SYSTEM(xxx)
 BIND ACQUIRE(USE) -
      ACTION(REPLACE) -
      CACHESIZE(1024) -
      CURRENTDATA(NO) -
      EXPLAIN(NO) -
      ISOLATION(CS) -
```

```
      LIBRARY('TCPIP.SEZADBRM') -
      MEMBER(EZAFTPMQ) -
      NODEFER(PREPARE) -
      PLAN(EZAFTPMQ) -
      RELEASE(COMMIT) -
      VALIDATE(RUN) -
      RETAIN
 END
//*
//FTPGRANT EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN=xxxxxx.DSNLOAD,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSTSIN  DD   *
 DSN  SYSTEM(xxx)
 RUN  PROGRAM(DSNTIAD) -
      PLAN(DSNTIAYY) -
      LIBRARY('xxxxxx.RUNLIB.LOAD')
 END
//SYSIN    DD   *
 GRANT EXECUTE ON PLAN EZAFTPMQ TO PUBLIC;
//*
```

Accessing DB2 Modules

The FTP server or client loads 3 DB2 modules into storage to perform an SQL query. These modules are:

- DSNALI
- DSNHLI2
- DSNTIAR

The modules are usually found in the DB2 load library with the suffix DSNLOAD. The DB2 administrator or system programmer should add the DSNLOAD library to the LINKLIST to ensure FTP has access to this library.

Another way to ensure access is to add the DSNLOAD library to the FTP STEPLIB. For the FTP server this means the JCL used to start the FTP server has a STEPLIB DD statement referring to the DSNLOAD library or, if the FTP daemon is started from the z/OS shell, the STEPLIB environment variable is set. For the FTP client, this means a TSO CLIST must allocate the DSNLOAD library as the STEPLIB.

If the FTP client is to be run from a batch job to perform SQL queries, the DSNLOAD library must be added to the STEPLIB DD statement for the batch job.

Usage Notes:

To allow FTP access to multiple levels of DB2, link to the libraries that contain the lowest level of DB2 to be accessed.

FTP.DATA Updates for SQL Query Function

To obtain FTP.DATA updates for the SQL query function, follow these steps:

1. Set the FTP.DATA DB2 statement to specify the name of the DB2 subsystem.
2. Set DB2PLAN to specify the DB2 plan to be used by the FTP server.
3. Set the SPREAD statement to specify whether SQL output is in spreadsheet format.
4. Set SQLCOL to specify the column headings of the output data.

Trivial File Transfer Protocol (TFTP)

TFTP is a TCP/IP protocol used to transfer files. TFTP can read or write files from or to a remote server. On the S/390 system, TFTP is a server you can configure with the command line option during TFTP invocation.

Considerations for z/OS

TFTP is installed in the /usr/lpp/tcpip/sbin/ directory.

Attention: The TFTP server uses well-known port 69. The TFTP server has no user authentication. Any client that can connect to port 69 on the server has access to TFTP. If the TFTP server is started without a directory, it allows access to the entire HFS. To restrict access to the HFS, start the TFTP server with a list of directories.

To start the TFTP server from the command line, type the tftpd command.

```
tftpd [-l] [-p port] [-t timeout] [-r maxretries] [-c concurrency_limit]
   [-s maxsegsize] [-f file] [-a archive directory [-a ...]]
   [directory ...]
```

The following are parameters used for the tftpd command:

-l Logs all incoming read and write requests and associated information to the system log. Logged information includes the IP address of the requestor, the file requested, and whether the request was successful.

-p port
Uses the specified port. The TFTP server usually receives requests on well-known port 69. You can specify the port in which requests are to be received.

-t timeout
Sets the packet timeout. The TFTP server usually waits 5 seconds before assuming a transmitted packet has been lost. You can specify a different timeout period in seconds.

-r maxretries
Sets the retry limit. The TFTP server usually limits the number of retransmissions it performs due to lost packet to 5. You can specify a different retry limit.

-c concurrency_limit
Sets the concurrency limit. The TFTP server spawns both threads and processes to handle incoming requests. You can specify the limit for the number of threads that may be concurrently processing requests under a single process. When the limit is exceeded, a new process is spawned to handle requests. The default is 200 threads.

-s maxsegsize
Sets the maximum block size that can be negotiated by the TFTP block size option. The default is 8192.

-f file Specifies a cache file. You can specify a file containing information on files to be preloaded and cached for transmission. A cache file consists of one or more entries. For clarity, place each entry on a separate line. An entry has the form:

a | b <pathname>

where:

- *a* indicates that the specified file is cached in ASCII form. The file is preconverted to netascii format.
- *b* indicates that the specified file is cached in binary form, with no conversion.

Following are examples of cache file entries,

```
a  /usr/local/textfile
b  local/binaryfile
```

If a relative pathname to the file is specified, the TFTP server searches the specified directories for the file.

The cached version of a file is only used for requests requiring the specified format. For example, the binary cached version of a file is not used in satisfying a request for the file in netascii format. If a file is to be retrieved in both binary and ASCII formats, the user must specify that two copies of the file be cached with one in binary format, and the other in netascii format.

Caching is not dynamic. The cache files are read in when the TFTP server is started and are not updated, even if the file on disk is updated. To update or refresh the cache, the TFTP server must be recycled.

-a archive directory
Specifies an archive directory. The files in this directory and its subdirectories are treated as binary files for downloading. This option is useful on EBCDIC machines that act as file servers for ASCII clients. Multiple -a options can be specified; one directory per -a option. Directories must be specified as absolute path names. You can specify no more than 20 directories.

directory
Specifies an absolute path name for a directory. You may specify no more than 20 directories on the tftpd command line.

If the TFTP server is started without a list of directories, all mounted directories are considered active.

If a list of directories is specified, only those directories specified are active. That list is used as a search path for incoming requests specifying a relative path name for a file.

Activating a directory activates all of its subdirectories.

For a file to be readable by the TFTP server, the file must be in an active directory and have world (″other″) read access enabled. For a file to be writable by the TFTP server, the file must already exist in an active directory and have world (″other″) write access.

The TFTP server preforks a child process to handle incoming requests when the concurrency limit is exceeded. Consequently, immediately after starting the TFTP server, two TFTP processes exist.

In case of a flood of concurrent TFTP commands, the TFTP server may fork additional processes. When the number of concurrent requests being processed drops below the concurrency limit, the number of TFTP processes is decreased back to two.

To terminate the TFTP server, send a SIGTERM signal to the oldest existing TFTP process. This is the process with a parent process ID of 1. Termination of this process will cause all of its children to terminate.

Verification of FTP

Verify Server

If FTP is in the autolog list and the TCP/IP address space is restarted, FTP should start automatically. For other cases, it should be started manually. To do this, go to the MVS Console and enter the following command:

```
S FTPD
```

Note: It is assumed that the FTP procedure is FTPD.

If the FTP server startup is complete, the following message should be seen on the MVS console:

```
EZY2702I Server-FTP: Initialization completed at 17:37:29 on  12/17/99.
```

If the message is not seen, a message explaining why FTP did not start up will appear in SYSLOG. Even if the above message is issued, it would be beneficial to inspect SYSLOG for warning messages issued during FTP initialization. EZY2700I displays the port FTP uses as the control port, the port it listens to for incoming connections from clients. In this example, FTP is listening to standard port 21.

The file syslog uses is defined in /etc/syslog.conf. The statement **daemon.info /tmp/daemon.log** directs SYSLOGD to save all the daemon messages in /tmp/daemon.log. Below is an example of output error messages.

```
EZYFT18I Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
EZY2697I IBM FTP CS V2R10 17:54:29 on 12/17/99
EZY2640I Using dd:SYSFTPD=USER1.PROFILES(FTPDHAC) for local site configuration parameters.
-->EZYFT46E Error in dd:SYSFTPD file: line 3 near column 16.
-->EZA2831E EMAILADDRCHECK value must be NO, WARNING or FAIL
-->EZYFT47I dd:SYSFTPD file, line 15: Ignoring keyword "ANONYMOUSFILEACCESS".
EZYFT21I Using catalog '/usr/lib/nls/msg/C/ftpdrply.cat' for FTP replies.
EZYFT26I Using 7-bit conversion derived from 'ISO8859-1' and 'IBM-1047' for the
EZYFT32I Using the same translate tables for the control and data connections.
EZYFT09I system information for MVSVIC97: OS/390 version 02 release 08.00 (9672)
EZYFT51I OS/390 version 2, release 08, modification 0000.
EZY2700I Using port FTP control (21)
EZY2701I Inactivity time is 0
-->EZY2658W domain name unknown: gethostbyname() error.
EZY2702I Server-FTP: Initialization completed at 17:54:30 on 12/17/99.
EZYFT41I Server-FTP: process id 16777232, server job name FTPD1
```

Verify Client

To verify that the FTP client works correctly, log onto TSO and issue the NETSTAT HOME command. This command will show the interface addresses that are known to the system. Below is the output of an example of the output from NETSTAT HOME:

```
NETSTAT HOME
 EZZ2350I MVS TCP/IP NETSTAT CS V2R10       TCPIP NAME: TCPCS        18:10:28
 EZZ2700I Home address list:
 EZZ2701I Address          Link             Flg
 EZZ2702I -------          ----             ---
```

```
 EZZ2703I 9.67.113.63      TR1              P
 EZZ2703I 9.67.1.1         LCTC1
 EZZ2703I 127.0.0.1        LOOPBACK
 READY
```

To invoke the FTP client, use any address shown on the NETSTAT HOME with the port shown on EZY2700I. In the example below, the client is invoked against 9.67.113.63. The environments shown below are TSO and Batch with only the output from the TSO displayed. Similar results would be expected when executing in either the USS or REXX environments.

```
//FTPBATCH JOB
//BATCH  EXEC PGM=FTP
//STEPLIB DD DSN=USER.LINKLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//STDOUT  DD SYSOUT=*
//STDERR  DD SYSOUT=*
//OUTPUT  DD SYSOUT=*
//INPUT DD *
  9.67.113.63
  USER4 <pw>
  SITE FILE=SEQ
  QUIT
//*

 READY
FTP 9.67.113.63
 IBM FTP CS V2R10  1999 349 01:35 UTC
 FTP: using TCPCS
 Connecting to:  9.67.113.63 port: 21.
 220-FTPD1 IBM FTP CS V2R10 at TEST, 18:15:03 on 1999-12-17.
 220 Connection will not timeout.
 NAME (9.67.113.63:USER4):
USER4
 >>> USER USER4
 331 Send password please.
 PASSWORD:

 >>> PASS
 230 USER4 is logged on.  Working directory is "/".
 Command:
```

Verify FTP.DATA Statements

Many FTP.DATA statements can be verified via the FTP client STAT and LOCSTAT commands. The output from each installations STAT and LOCSTAT will depend on the client and server copy of FTP.DATA. Below is sample output of one system.

```
stat
 EZA1701I >>> STAT
 211-Server FTP talking to host 9.67.113.63, port 1030
 211-User: USER1  Working directory: /tmp
 211-The control connection has transferred 113 bytes
 211-There is no current data connection.
 211-The next data connection will be actively opened
 211-to host 9.67.113.63, port 1030,
 211-using Mode Stream, Structure File, type ASCII, byte-size 8
 211-Automatic recall of migrated data sets.
 211-Automatic mount of direct access volumes.
 211-Auto tape mount is allowed.
 211-Inactivity timer is disabled
 211-VCOUNT is 50
 211-ASA control characters in ASA files opened for text processing
 211-will be transferred as ASA control characters.
 211-Trailing blanks are removed from a fixed format
 211-data set when it is retrieved.
```

```
 211-Data set mode.  (Do not treat each qualifier as a directory.)
 211-Primary allocation 1 track.  Secondary allocation 1 track.
 211-Partitioned data sets will be created with 27 directory blocks.
 211-FileType JES (MVS Job Spool). JES Name is JESE
 211-Number of access method buffers is 5
 211-RDWs from variable format data sets are discarded.
 211-Site DB2 subsystem name is DB2
 211-Data not wrapped into next record.
 211-JESLRECL is 80
 211-JESRECFM is Fixed
 211-JESINTERFACELEVEL is 2
 211-JESENTRYLIMIT is 10
 211-JESOWNER is USER1
 211-JESJOBNAME is USER1*
 211-JESSTATUS is ALL
 211-SMS is active.
 211-New data sets will be catalogued if a store operation terminates abnormally
 211-Single quotes will override the current working directory.
 211-UMASK value is 027
 211-Process id is 16777227
 211-Checkpoint interval is 0
 211-Record format VB, LRECL: 256, Blocksize: 6233
 211 *** end of status ***
 EZA1460I Command:

locstat
 EZA1600I Trace: FALSE, Send Port: TRUE
 EZA1601I Send Site with Put command: TRUE
 EZA2677I Connected to:9.67.113.63, Port: FTP control (21), not logged in
 EZA1605I Local Port: 1030
 EZA1606I Data type:a, Transfer mode:s, Structure:f
 EZA2098I Automatic recall of migrated data sets.
 EZA2100I Automatic mount of direct access volumes.
 EZA2101I Data set mode. (Do not treat each qualifier as a directory.)
 EZA2137I Primary allocation 1 track, Secondary allocation 1 track.
 EZA2138I Partitioned data sets will be created with 27 directory blocks
 EZA2103I FileType is SEQ (Sequential - the default).
 EZA2141I Number of access method buffers is 5.
 EZA2145I RDW's from VB/VBS files are discarded.
 EZA2148I DB2 subsystem name is DB2
 EZA2152I Volid of Migrated Data Sets is MIGRAT
 EZA2154I Trailing blanks in records read from RECFM F datasets are discarded.
 EZA2535I Record format: VB, LRECL: 256, Blocksize: 6233.
 EZA2801I Data not wrapped into next record.
 EZA2494I Checkpoint interval is 0
 EZA2425I RESTGet          Checkpoint data set will be opened for get.
 EZA2816I No automatic mount of tape volumes.
 EZA2809I CCONNTIME is 30
 EZA2810I DATACTTIME is 120
 EZA2811I DCONNTIME is 120
 EZA2812I INACTTIME is 120
 EZA2813I MYOPENTIME is 60
 EZA2815I VCOUNT is 50
 EZA2689I Prompting: ON, Globbing: ON
 EZA2719I ASA control characters transferred as ASA control characters
 EZA2720I New data sets catalogued if a store operation terminates abnormally
 EZA2722I Single quotes will override the current working directory
 EZA2724I UMASK value is 027
 EZY2640I Using 'USER1.FTP.DATA' for local site configuration parameters.
 EZA1460I Command:
```

Verifying Anonymous, Banner, and Other Optional Configuration Information

Depending on your installations' choices for anonymous level, banner support chosen, exits, etc., verification of support output will differ. To verify anonymous

configuration at a particular installation, log in as anonymous and verify the behavior is as expected. For example, if EMAILADDRCHECK FAIL is specified in FTP.DATA, try to log in as anonymous using an invalid e-mail address as password. To verify banner support, login and verify the banners are displayed as expected. Below is a sample of FTP.DATA and FTP client output for one such installation.

```
; BANNER STUFF
EMAILADDRCHECK YES
BANNER USER1.TEST1
ADMINEMAILADDR FTPADMIN@MYSYSTEM.COM
; ANONYMOUS STUFF
ANONYMOUSLEVEL 1
ANONYMOUS USER1
STARTDIRECTORY HFS
```

```
ftp 9.67.113.63
 IBM FTP CS V2R10  1999 349 01:35 UTC
 FTP: using TCPCS
 Connecting to:  9.67.113.63 port: 21.
 220-FTPD1 IBM FTP CS V2R10 at CRAP, 19:02:45 on 1999-12-17.
 220-You have just read 'USER1.TEST1'
 220-ADMINEMAILADDRESS is FTPADMIN@MYSYSTEM.COM
 220 Connection will not timeout.
 NAME (9.67.113.63:USER4):
anonymous no-email-pw
 >>> USER anonymous
 331 Send password please.
 >>> PASS
 550 PASS command failed - __passwd() error : EDC5111I Permission denied.
 Command:
```

Verify FTP-JES Interface (Optional)

As with the other optional configuration information, FTP-JES support can best be verified by logging in and confirming the FTP.DATA parameters chosen. To verify JES support, a simple batch job can be created if the JESINTERFACELEVEL is set to the security requirements of an installation. Below is the batch job and FTP client output for JESINTERFACELEVEL 2.

```
EDIT       USER1.FTP.JCL.TEST                              Columns 00001 00072
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100 //JOBTEST  JOB MSGCLASS=H,MSGLEVEL=(1,1),CLASS=A,
000200 //         USER=USER1,PASSWORD=TCPSUP
000300 //STEP1    EXEC PGM=IEBGENER
000400 //OBJTMP1  DD DSN=&PRLOBJ,DISP=(NEW,PASS,DELETE),
000500 //            SPACE=(CYL,(1,1,10)),
000600 //            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
000700 //SYSPRINT DD SYSOUT=A
000800 //SYSUT1   DD DSN=SYS1.PROCLIB(JES2),DISP=SHR
000900 //SYSIN    DD DUMMY
001000 //SYSUT2   DD SYSOUT=H
001100 //
001200 //         EXEC PGM=IEFBR14
****** **************************** Bottom of Data ****************************
```

```
site file=jes jesjobname=jobtest jesowner=* jesstatus=all
 EZA1701I >>> SITE file=jes jesjobname=jobtest jesowner=* jesstatus=all
 200 Site command was accepted
 EZA1460I Command:
put 'user1.ftp.jcl.test'
 EZA1701I >>> SITE FIXrecfm 80 LRECL=80 RECFM=FB BLKSIZE=32720
 200 Site command was accepted
 EZA1701I >>> PORT 127,0,0,1,4,12
 200 Port request OK.
 EZA1701I >>> STOR 'user1.ftp.jcl.test'
 125 Sending Job to JES internal reader FIXrecfm 80
```

```
 250-It is known to JES as JOB00076
 250 Transfer completed successfully.
 EZA1617I 984 bytes transferred in 0.005 seconds.  Transfer rate 196.80 Kbytes/sec.
 EZA1460I Command:
dir j76
 EZA1701I >>> PORT 127,0,0,1,4,13
 200 Port request OK.
 EZA1701I >>> LIST j76
 125 List started OK for JESJOBNAME=JOBTEST, JESSTATUS=ALL and JESOWNER=*
 EZA2284I JOBNAME  JOBID    OWNER    STATUS CLASS
 EZA2284I JOBTEST  JOB00076 USER1    OUTPUT A        RC=000
 EZA2284I          ID  STEPNAME PROCSTEP C DDNAME   BYTE-COUNT
 EZA2284I          001 JESE              H JESMSGLG      1084
 EZA2284I          002 JESE              H JESJCL        1023
 EZA2284I          003 JESE              H JESYSMSG      1143
 EZA2284I          004 STEP1             H SYSUT2         741
 EZA2284I          005 STEP1             A SYSPRINT       209
 EZA2284I 5 spool files
 250 List completed successfully.
 EZA1460I Command:
```

Chapter 8. Domain Name System (DNS)

This chapter contains information about configuring the name server in a BIND-based Domain Name System (DNS). BIND (Berkeley Internet Name Domain) is the most common implementation of a DNS. BIND was developed at the University of California, Berkeley and is currently maintained by the Internet Software Consortium (ISC). The name server is based on BIND 4.9.3.

This chapter also contains information about connection optimization, which uses DNS for distributing connections among hosts or server applications within a sysplex domain.

The Domain Name System is a client/server model in which programs called *name servers* contain information about host systems and IP addresses. Name servers provide this information to clients called *resolvers*.

This chapter is not intended to be a comprehensive description of DNS or of BIND. For more complete descriptions, refer to texts such as *DNS and BIND, 3rd Edition* by Paul Albitz and Cricket Liu (O'Reilly & Associates, Inc., 1997).

DNS and BIND Overview

While TCP/IP applications refer to host computers by their IP addresses, it easier to use host names. To enable the use of host names in a network, the Domain Name System (DNS) translates host names to IP addresses. DNS provides the host name-to-IP address mapping through network server hosts called *domain name servers*. For detailed information about name servers, see "Domain Name Servers" on page 330. DNS can also provide other information about server hosts and networks such as the TCP/IP services available at a server host and the location of domain name servers in a network.

DNS organizes the hosts in a network into domains. A *domain* is a group of hosts that share the same name space in the domain hierarchy and are usually controlled within the same organization. Domains are arranged in a hierarchy. A special domain known as the *root domain* exists at the top of the hierarchy. The root domain servers store information about server hosts in the root domain and the name servers in the delegated, *top-level* domains, such as *com* (commercial), *edu* (education), and *mil* (military). The name servers in the top-level domain, in turn, store the names of name servers for their delegated domains, and so on.

The complete name of a host, also known as the *fully qualified domain name* (FQDN), is a series of labels separated by dots or periods. Each label represents an increasingly higher domain level within a network. The complete name of a host connected to one of the larger networks generally has more than one subdomain, as shown in the following examples:

```
host1.subdomain2a.subdomain2.rootdomain
user4720.eng.mit.edu
```

A domain name server requires the FQDN. The resolver combines the host name with the domain name to create the FQDN before sending the name resolution request to the domain name server.

DNS also provides IP address-to-host name mapping using a special domain called *in-addr.arpa*. This kind of mapping is useful for producing output (host names) that is easy to read. The format of an in-addr.arpa name is the reverse octet of an IP

address concatenated with in-addr.arpa. For example, the address 9.67.30.143 has an in-addr.arpa name of 143.30.67.9.in-addr.arpa.

A system administrator can name the host systems and domains in the local, private network with any name you want, but to link with name servers in a public network like the Internet, you need to determine which domain you want to be in (which parent domain) and then contact the registrar in that domain to register the names and IP addresses of your name servers. This ensures that queries from outside the domain being defined can be answered by this name server if need be.

Note: Contact the InterNetwork Information Center (InterNIC) for more information about Internet registration. You can contact InterNIC by pointing your Web browser at http://www.internic.net.

Domain Name Servers

A name server is said to be *authoritative* for some part of the domain name space, called a *zone*. A zone consists of the resources within a single domain (for example, commercial or `.com`) or subdomain (for example, `raleigh.ibm.com`). Typically, a zone is administered by a single organization or individual.

All host systems in a given zone share the same higher level domain name (for example, `host1.raleigh.ibm.com`, `host2.raleigh.ibm.com`, `host3.raleigh.ibm.com`, and so on). As system administrator, you create a zone of authority by listing all the host systems in your zone in the database file of the name server that is authoritative for the zone.

If a domain name server receives a query about a host for which it has information in its database or in its cache, it performs the name resolution and returns all the address records associated with the host to the client. Some hosts (for example, routers or gateways between two or more networks) might have more than one IP address.

Alternatively, the name server can query other name servers for information. This process is called *iterative resolution*. The local name server successively queries other name servers, each of which responds by referring the local name server to a remote name server that is closer to the name server authoritative for the target domain. Finally, the local name server queries the authoritative name server and gets an answer. If the information about a requested host name does not exist or if a name server does not know where to go for the information, it sends a negative response back to the client.

There are two kinds of name servers in the DNS:

- Master servers (primary and secondary)
- Caching-only servers

A single server can perform multiple functions. For example, it can be a primary server and a secondary server for different zones. The purpose of having these different kinds of servers is to provide redundancy (in case of system failure), to distribute the workload among multiple servers, to speed up the name-resolution process, and to provide flexibility in network design. In addition to being a master or caching-only server, a name server can be defined to only contact a specific set of name servers if queries cannot be resolved locally (through the use of forwarders).

The following sections discuss master servers, caching-only servers, and forwarding.

Master Servers

A master server is the authority for its domain. It queries and is queried by other name servers in the DNS. The data it receives in response from other name servers is cached. Master servers are not authoritative for cached data.

There are two types of master servers: primary and secondary. Each domain must have only one primary name server, and it should have at least one secondary name server for backup. Calling a *particular* name server a primary or secondary is misleading. Any given name server can take on either or both roles, as defined by the boot file.

Primary Name Servers: A primary name server maintains all the data for its zone. Static resources are kept in database files called *domain data files*. For information on creating domain data files, see "Step 5. Create the Domain Data Files (Primary Name Server Only)" on page 336. Primary name servers can also receive zone updates dynamically. For information on dynamic DNS, see "Dynamic IP" on page 372. For information on dynamic generation of resources, see "Connection Optimization in a Sysplex Domain" on page 358.

Secondary Name Servers: A secondary name server acts as an alternate to the primary server if the primary name server becomes unavailable or overloaded. The secondary name server receives zone data directly from the primary name server in a process called *zone transfer*. Zone transfers, which only occur when data has changed, are based on the refresh interval in the Start of Authority (SOA) resource record. For a description of the SOA resource record, see *z/OS Communications Server: IP Configuration Reference*. A secondary server, like a primary server, is authoritative for a domain.

Caching-Only Servers

All name servers cache (store) the data they receive in response to a query. A caching-only server, however, is not authoritative for any domain. When a caching-only server receives a query, it checks its cache for the requested information. If it does not have the information, it queries a local name server or a root name server, passes the information to the client, and caches the answer for future queries. The names and addresses of root name servers are stored in its hints (root server) file, the name and file path of which are specified in the name server's boot file.

You can use caching servers to create a large cache of responses to frequently requested queries and reduce the number of queries made to master servers. The caching server stores data for a period of time determined by the time-to-live (ttl) value, and the cached information is lost if the name server is restarted.

Forwarders

Normally, name servers answer queries from cached data or, if that does not succeed, they attempt to contact other name servers identified in their data files as authoritative for certain domains. However, name servers can also be configured to contact special servers called *forwarders* before contacting the name servers listed in their data files. If a forwarder cannot process the query and if the local name server is not a forward-only name server, the local name server contacts the name servers in its data files. A forward-only name server relies completely on its forwarders, it does not try to contact other servers to find out information if the forwarders do not give it an answer.

The forwarding function is useful for reducing the number of queries to servers on the Internet and for creating a large cache of information on forwarders. It is also a useful function for providing Internet access for local servers that, for one reason or another, do not have access themselves.

Recommended Reading

In addition to books such as *DNS and BIND, 3rd Edition* by Paul Albitz and Cricket Liu (O'Reilly & Associates, Inc., 1997), you can also read the *Name Server Operations Guide for BIND* available from the ISC. To obtain copies of the guide, point your browser to http://www.isc.org/isc/bind.html.

For additional information on DNS in a sysplex, refer to the following Redbooks:

- z/OS eNetwork™ Communications Server V2R7 TCP/IP Implementation Guide Volume 2: UNIX Applications, SG24-5228
- TCP/IP in a Sysplex, SG24-5235-01

For information on Dynamic IP, see "Dynamic IP" on page 372.

If you wish to request participation in a mail group that discusses issues related to BIND, contact `bind-request@uunet.uu.net`.

DNS protocols are described in various Request for Comments (RFC) papers and Internet drafts. RFCs outline existing protocols, suggest new protocols, and establish standards for the Internet protocol suite. Internet drafts are proposals, techniques, and mechanisms that document Internet Engineering Task Force (IETF) work-in-progress.

For information about obtaining RFCs, see "Appendix B. Related Protocol Specifications" on page 551.

The following three RFCs contain basic information about the DNS:

1033 *Domain Administrators Operations Guide*, M. Lottor

1034 *Domain Names—Concepts and Facilities*, P.V. Mockapetris

1035 *Domain Names—Implementation and Specification*, P.V. Mockapetris

Setting Up and Running the Name Server

This section describes the tasks involved in configuring the name server and verifying that the name server is working correctly.

Name server configuration files are arranged in a Hierarchical File System (HFS). Before configuring DNS, the TSO user ID from which the name server is started must have the proper authority to access the name server boot and zone files. For a complete description of file permissions within the HFS, refer to *z/OS UNIX System Services Planning*.

Configuring a Primary Name Server

The name resolution process is an example of a client/server relationship in which clients, through their resolvers, request a service (name resolution) from name servers. For a general overview of name servers, see "Domain Name Servers" on page 330.

The following summary lists the steps for configuring a master server or a caching-only server:

1. Create the boot file.
2. Specify stack affinity (multiple stack environment).
3. Specify port ownership.
4. Update the name server start procedure.
5. Create the domain data files (primary name server only).
6. Create the hints (root server) file.
7. Create the loopback file.
8. Ensure that the syslog daemon is running on your system.
9. Specify whether the name server is to run swappable or nonswappable.
10. Start the name server.
11. Verify that the name server started correctly.
12. Verify that the name server can accept queries.

The difference between the configuring a primary name server and secondary and caching-only servers is the creation of domain data files (the database files containing host-to-address and address-to-host mappings). The domain data files are maintained on the primary name server, and the secondary name server transfers this data to its own database. Examples of secondary, caching-only, and forward-only configurations are in “Configuring a Secondary Name Server” on page 344, “Configuring a Cache-only Name Server” on page 346, and “Adding Forwarding to Your Name Server” on page 347.

Step 1. Create the Boot File

The boot file is the main configuration file for a domain name server. The named daemon reads the boot file for information about how to set up the local name server. The records in the boot file identify the type of name server, the zones over which it has authority, the location of data for setting up its name resolution database, and other configuration options. The default name of the boot file is `/etc/named.boot`. You can specify an alternate boot file using the `-b` named start option. For information about named options, refer to *z/OS Communications Server: IP Configuration Reference*.

Note: The named daemon reads the boot file only when the named daemon starts or when it receives a SIGHUP signal. For a description of named signals, refer to *z/OS Communications Server: IP Configuration Reference*.

Each type of name server has a special boot file configuration. You create a boot file using directives. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Boot files created locally for use by the name server are assumed to be in code page IBM-1047. For systems using other code pages, use the `iconv` command to translate from the local code page to code page IBM-1047. See *z/OS UNIX System Services Command Reference* for more information.

A boot file for a primary name server (a name server that maintains all the data for its zone in database files) will need, at a minimum, to specify the zones for which the name server will be authoritative, their locations in the hfs, and the location of the hints file (the location of root name servers). A loopback file is also recommended.

The sample boot file shipped in /usr/lpp/tcpip/samples/named.boot is shown below. Refer to the program directory for its location.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;             5647-A01 (C) COPYRIGHT IBM CORP. 1997
;
; COMPONENT_NAME: TCPIP named.boot
;
; FUNCTIONS:
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUP
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;
;  /etc/named.boot
;
;        boot file for name server
;
;
; type        domain                source file or host
;
directory     /etc/dnsdata
primary       mycorp.com            named.for
primary       34.37.9.in-addr.arpa   named.rev
primary       0.0.127.in-addr.arpa   named.lbk
cache         .                      named.ca
options       query-log
```

This boot file specifies:

- The location of the files (/etc/dnsdata)
- That the name server will be the primary name server for the `mycorp.com` zone
- That the data for mycorp.com is contained in /etc/dnsdata/named.for
- That the name server will be the primary name server for the reverse mapping zone, `34.37.9.in-addr.arpa`
- That the data for addresses `9.37.34.x` contained in the zone will be specified in /etc/dnsdata/named.rev
- That the list of root name servers is in /etc/dnsdata/named.ca
- That the name server defines the loopback address in the 0.0.127.in-appr.arpa zone and the data is contained in the file, /etc/dnsdata/named.lbk
- That all queries coming in to this name server will be logged in the syslog daemon output file

Step 2. Specify Stack Affinity (Multiple Stack Environment)

In a multiple stack environment, the names server is like any application. It binds to the stack specified by TCPIPJOBNAME. To run multiple name servers, each must use a different TCPIPJOBNAME. See “Considerations for Multiple Instances of TCP/IP” on page 37 for more information about specifying TCPIPJOBNAME.

Note: When changing TCPIPJOBNAME, any client needing to be connected to the new stack name must be restarted (for example, any application, including the name server, that is currently running, that is bound to the new stack name specified by TCPIPJOBNAME).

Step 3. Specify Port Ownership

The name server uses a single port (53) for TCP and UDP sessions. To specify port ownership when using the named start procedure or when starting the name server directly from z/OS UNIX, add the following statements to the PROFILE.TCPIP data set:

```
PORT
  53 TCP NAMED
  53 UDP NAMED
```

For more information on the PORT statement, refer to *z/OS Communications Server: IP Configuration Reference*.

Note: In order to pick up changes in the PROFILE.TCPIP data set, stop and restart TCP/IP. As an alternative to stopping the stack, use the VARY OBEY command to reserve the ports while the stack is up.

Step 4. Update the Name Server Start Procedure (Optional)

When choosing to start the name server from MVS, create a started procedure. This is not necessary if the name server is started from the z/OS UNIX shell. Move the sample started procedure, `SEZAINST(NAMED)`, to a recognized PROCLIB. Specify name server parameters and change the data set names as required to suit local configuration. The boot file path can also be changed, which in the following sample start procedure is `/etc/named.boot`. If you want to have NAMED messages written out to SYSLOGD (HFS file) instead of the system console (syslog), then you must start NAMED via BPXBATCH as shown below:

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZANSPRO
//*
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        5647-A01 (C) Copyright IBM Corp. 1997, 2000.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//*        NAMED can be started with a variety of parameters.
//*        In this example, the "-b" parameter describes which
//*        boot file NAMED should be started with.
//*
//NAMED    PROC B='/etc/named.boot'
//NAMED    EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
//         PARM='PGM /usr/lpp/tcpip/sbin/named -b &B '
//*
//*        NAMED can use certain environmental variables, such
//*        as NLSPATH (to determine the location of the message
//*        catalog), and RESOLVER_CONFIG (to determine the location
```

```
//*        of the file that contains the parameter TCPIPjobname).
//*        These variables can be specified in a file defined
//*        by STDENV.
//*        An example of the contents of this file follows:
//*
//*             RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*                          or
//*             RESOLVER_CONFIG=/etc/resolv.conf.tcp2
//*
//*        Define STDENV with the name of the file that contains
//*        the environmental variables to be used for this
//*        invocation of NAMED.
//*
//*STDENV   DD PATH='/etc/named.env',
//*            PATHOPTS=(ORDONLY)
//*STDENV  DD DSN=SAMPLE.NAMED(ENV&SYSCLONE),DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
```

Step 5. Create the Domain Data Files (Primary Name Server Only)

The domain data files contain information about a domain, such as the IP addresses and names of the hosts in the domain for which the primary name server is authoritative. The *forward* domain data file contains entries that provide forward mapping (host names-to-IP addresses for each host system in the zone) as well as additional information about system resources. The *reverse* domain data file contains entries that provide reverse mapping (IP addresses-to-host names). A separate reverse domain data file for each network (or subnet) in a domain can be created. Definition of WLM and dynamic zones is covered in “Advanced Name Server Topics” on page 358.

Note: The TSO user ID from which the name server is started must have the proper authority to access the name server boot and zone files. For a complete description of file permissions within the HFS, see the *z/OS UNIX System Services Planning* (SC28–1890–02).

Domain data files can be named anything, but for convenience in maintaining the named database, give them names of the form `named.extension`, where the extension identifies the type of file. This book uses the extension `.rev` to specify the reverse domain data file, `.for` to specify the forward domain data file, and `.bak` to specify a backup file.

Use the following to create domain data files:

- Control entries
- Resource records
- Special characters

Note: Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these files.

The sample forward domain file SEZASAMP(DB) is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/named.for.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;             5647-A01 (C) COPYRIGHT IBM CORP. 1997, 2000
;
; COMPONENT_NAME: TCPIP named.data
```

```
;
; FUNCTIONS: nameserver
;
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1989, 2000
; All Rights Reserved
; Licensed Materials - Property of IBM
;
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;

;   /etc/named.for file
     ;
     $ORIGIN ibm.com.
     raleigh    IN  SOA  buzz1.raleigh.ibm.com. bug@vmail (                 {1}
        1       ; Serial--  change when database is changed
        10800   ; Refresh-- secondary checks every three hours
        3600    ; Retry--   secondary retries connection
                ;            every hour after a failed zone transfer
        604800  ; Expire--  data expires after one week
        86400 ) ; Time to Live-- data cached in other servers one day)
     ;
     $ORIGIN raleigh.ibm.com.                                              {2}
                  IN      NS      buzz1                                    {3}
                  IN      NS      elmer
                  IN      NS      jumbo
     _http._tcp           SRV     0  0 80   buzz1.ibm.com.                {4}
                          SRV     10 0 8000 elmer.ibm.com.                 {4}
     _http._tcp.www       SRV     0  0 80   buzz1.ibm.com.                {5}
                          SRV     10 0 8000 elmer.ibm.com.                 {5}
     ; OWNER     CLASS   TYPE      RECORD DATA
     localhost    IN      A       127.0.0.1
     buzz1        IN      A       9.37.34.149
     elmer        IN      A       9.37.34.7
     jumbo        IN      A       8.37.34.12
     dog          IN      A       8.37.34.113                              {6}
                  IN      A       9.37.34.113
     vmail        IN      A       7.37.34.1
     kent         IN      A       7.37.34.2
     greg         IN      A       7.37.34.3
     printserver  IN      A       7.37.34.4
                  IN      A       8.37.34.33
                  IN      A       9.37.34.44
     gary         IN      CNAME   jumbo                                    {7}
```

{1}
The SOA (Start of Authority) record specifies the name server buzz1 as the authoritative name server for the domain raleigh.ibm.com. The mail address of the person responsible for domain data is bug@vmail. The numbers enclosed in parentheses are parameters used primarily to manage the zone information.

```
{2}
     The control entry $ORIGIN appends the string raleigh.ibm.com. to all the following host
names that do not end with a
dot ('.').

{3}
     The NS (Name Server) records specify the name servers in the zone. Note that NS records
do not distinguish between
primary and secondary name servers.

{4}
 The SRV records specify the location for the 'http' service using the 'tcp' protocol.
The first record has a priority of 0,
a weight of 0, uses port 80 and the service is provided at host, buzz1.ibm.com.  The second
record has a priority of 10 which
is lower, a different port and target.  A web client capable of using SRV records requesting
http://ibm.com/ would be directed to buzz1.ibm.com and elmer.ibm.com.  The client would be
responsible for
determining which site to connect to first based first on priority and then on weight.

{5}
 The SRV record also specifies the location for the 'http' service using the 'tcp' protocol.
 A web client capable of
using SRV records requesting http://www.ibm.com/ would also be directed buzz1.ibm.com and
lmer.ibm.com.

{6}
     This A (Address) record maps the host name (dog.raleigh.ibm.com) to the IP addresses of
he two networks to which
it is connected.

{7}
     The CNAME record specifies that the name gary is an alias for the host name
umbo.raleigh.ibm.com.
```

The sample reverse domain file SEZASAMP(REV) is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/named.rev.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;             5647-A01 (C) COPYRIGHT IBM CORP. 1997
;   /etc/dnsdata/named.rev
$ORIGIN 37.9.in-addr.arpa.
34 IN SOA buzz1.mycorp.com bug@mycorp.com. (
              1  10800  3600 604800 86400 )
              IN      NS     buzz1.mycorp.com.
              IN      NS     elmer.mycorp.com.
$ORIGIN 34.37.9.in-addr.arpa.
1             IN      PTR    vmail.mycorp.com.
4             IN      PTR    printserver.mycorp.com.
7             IN      PTR    elmer.mycorp.com.
12            IN      PTR    jumbo.mycorp.com
33            IN      PTR    printserver.mycorp.com.
44            IN      PTR    printserver.mycorp.com.
149           IN      PTR    buzz1.mycorp.com.
113           IN      PTR    dog.mycorp.com.
114           IN      PTR    dog.mycorp.com.
```

Note: Data files created locally for use by the name server are assumed to be in code page IBM-1047. For systems using other code pages, use the `iconv` command to translate from the local code page to code page IBM-1047. Refer to *z/OS UNIX System Services Command Reference* for more detailed information about this command. Files read through a network connection (for example, secondary data files) are converted to IBM-1047 by the name server before they are written to the local file system.

FTP can also be used to convert the files to code page IBM-1047.

Step 6. Create the Hints (Root Server) File

The hints file contains the names and IP addresses of the authoritative root domain name servers. The root name servers contain the names of name servers in the top-level domains such as com, edu, and mil. The name server uses root server information when deciding which name server to contact when it receives a query for a host outside its zone of authority and it does not have the data in its cache.

Note: The hints file does not contain cached data nor does the name server provide other hosts with the information contained in the hints file. A forward-only server is the only type of name server that does not require a hints file.

To obtain a hints file, point your Web browser at `ftp://ftp.rs.internic.net` and retrieve the file `named.root` from the `domain` subdirectory. Update your hints file on a regular basis.

The `cache` directive in a boot file specifies the path and name of the hints file. This guide uses the extension `.ca` to specify the hints file.

An example of a hints file originally copied from `ftp://ftp.rs.internic.net/domain/named.ca` is listed below. Continuing the example that began in the boot file, the file would be /etc/dnsdata/named.ca.

```
;   This file holds the information on root name servers needed to
;   initialize cache of Internet domain name servers
;   (e.g., reference this file in the "cache   .   <file>"
;   configuration file of BIND domain name servers).
;
;   This file is made available by InterNIC registration services
;   under anonymous FTP as
;   file                    /domain/named.root
;   on server               FTP.RS.INTERNIC.NET
;   -OR- under Gopher** at RS.INTERNIC.NET
;       under menu          InterNIC Registration Services (NSI)
;         submenu           InterNIC Registration Archives
;       file                named.root
;
;   last update:       May 19, 1997
;   related version of root zone:      1997051700
;
;   formerly NS.INTERNIC.NET
;
.                       3600000   IN  NS   A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000       A    198.41.0.4
;
;   formerly NS1.ISI.EDU
;
.                       3600000       NS   B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000       A    128.9.0.107
;
;   formerly C.PSI.NET
;
.                       3600000       NS   C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000       A    192.33.4.12
;
;   formerly TERP.UMD.EDU
;
.                       3600000       NS   D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000       A    128.8.10.90
;
;   formerly NS.NASA.GOV
```

```
;
.                       3600000       NS   E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000       A    192.203.230.10
;
;  formerly. NS.ISC.ORG
;
.                       3600000       NS   F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000       A    192.5.5.241
;
;   formerly NS.NIC.DDN.MIL
;
.                       3600000       NS   G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000       A    192.112.36.4
;
;   formerly AOS.ARL.ARMY.MIL
;
.                       3600000       NS   H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000       A    128.63.2.53
;
;   formerly NIC.NORDU.NET
;
.                       3600000       NS   I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000       A    192.36.148.17
;
;   temporarily housed at NSI (InterNIC)
;
.                       3600000       NS   J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000       A    198.41.0.10
;
;   housed in LINX, operated by RIPE NCC
;
.                       3600000       NS   K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000       A    193.0.14.129
;
;   temporarily housed at ISI (IANA)
;
.                       3600000       NS   L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000       A    198.32.64.12
;
;   temporarily housed at ISI (IANA)
;
.                       3600000       NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000       A    198.32.65.12
; End of File
```

Step 7. Create the Loopback File

The loopback file contains the loopback address. This is the address that a host uses to route queries to itself. The preferred loopback address is `127.0.0.1`, although you can configure additional loopback interfaces in the PROFILE.TCPIP. DNS will bind to `127.0.0.1` in addition to the first loopback address configured in PROFILE.TCPIP.

To get the sysplex domain name, add the following PTR record to the loopback file in the loopback zone:

127.0.0.128.in-addr.arpa. IN PTR Sysplex_Domain_Name.

Sysplex_Domain_Name is the domain name of the sysplex (specified as cluster zone in the primary boot file). Do not forget to put a period (.) after the Sysplex_Domain_Name. Note that all of the sysplex name servers must be updated with this change.

This guide uses the extension `.lbk` to specify the loopback file.

Note: In addition to creating the loopback file, add an address resource record called *localhost* to the forward domain data file. This record supports proper two-way resolution. The “Sample Forward Domain Data File” on page 354 contains a localhost record.

Use the following elements to create the loopback file:

- Control entries
- Resource records
- Special characters

Note: Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these files.

The sample loopback file shipped in SEZASAMP(LBK) is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/named.lbk.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;             5647-A01 (C) COPYRIGHT IBM CORP. 1997
;   /etc/dnsdata/named.lbk
0.0.127.in-addr.arpa. IN SOA  buzz1.mycorp.com. bug@vmail (
     1
     10800
     3600
     604800
     86400   )
0.0.127.in-addr.arpa.   IN   NS  buzz1.mycorp.com.
0.0.127.in-addr.arpa.   IN   NS  elmer.mycorp.com.
0.0.127.in-addr.arpa.   IN   NS  jumbo.mycorp.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
```

Step 8. Ensure that the Syslog Daemon is Running on Your System

The name server uses the syslog daemon to log messages. To verify that the name server starts correctly or to diagnose problems, the syslog daemon should be running.

If your syslog daemon is not configured, see “Creating the Syslog File” on page 348 for information regarding the syslog daemon.

Step 9. RACF-Authorize User IDs

You might want to have the name server run in a swappable state as it has in the past. This is an optional step. Keep in mind that when an application makes an address space nonswappable, it might cause additional real storage in the system to be converted to preferred storage. Because preferred storage cannot be configured offline, allowing the name server to run in a nonswappable state can reduce the installation’s ability to reconfigure storage in the future.

If you want to run the name server as swappable, you must have the ″BPX.STOR.SWAP″ FACILITY class profile defined to RACF with no universal access. To do this, enter the following commands from a RACF user ID.

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
SETROPTS RACLIST(FACILITY) REFRESH
```

The name server will also run as swappable if the ″BPX.STOR.SWAP″ FACILITY is not defined and the name server is started from a user ID with a UID not equal to 0.

If you want to run the name server in a nonswappable state, either:

- Do not define the BPX.STOR.SWAP facility to RACF and start the name server from a user ID with a UID=0.
- Define the facility to RACF and allow the appropriate users at least READ access to the facility.

The latter method can be accomplished with the following set of commands:

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
PERMIT BPX.STOR.SWAP CLASS(FACILITY) ID(userid) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Step 10: Specify whether the Name Server is to Run Swappable or Nonswappable

You might want to run the name server in a swappable state, as it has in the past. This is an optional step. Keep in mind that when an application makes an address space nonswappable, it might convert additional real storage in the system to preferred storage. Because preferred storage cannot be configured offline, allowing the name server to run in a nonswappable state can reduce the installation's ability to reconfigure storage in the future.

If you want to run the name server run as swappable, you must have the ″BPX.STOR.SWAP″ FACILITY class profile defined to RACF with no universal access. To do this, enter the following commands from a RACF user ID.

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
SETROPTS RACLIST(FACILITY) REFRESH
```

The name server will also run as swappable if the ″BPX.STOR.SWAP″ FACILITY is not defined and the name server is started from a user ID with a UID not equal to 0.

If you want the name server to run in a nonswappable state, either:

- Do not define the BPX.STOR.SWAP facility to RACF and start the name server from a user ID with a UID=0
- Define the facility to RACF and allow the appropriate users at least READ access to the facility.

The latter method can be accomplished with the following set of commands:

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
PERMIT BPX.STOR.SWAP CLASS(FACILITY) ID(userid) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Step 11. Start the Name Server

Your name server is ready to start. to Start the name server using the following methods:

- A supervisor with an authorized TSO ID can start a name server from the MVS operator's console by starting the named start procedure. If the boot file path is not `/etc`, specify the correct path in the start procedure. Refer to *z/OS Communications Server: IP Configuration Reference*. A sample start procedure is provided with the product and is found in SEZAINST(NAMED).
- A user ID with superuser authority can start the name server from the shell by starting z/OS UNIX, then issuing the `named` command and, optionally, any parameters.
- It is also possible to start the server automatically when z/OS UNIX is started by specifying the path and file name of the z/OS UNIX initialization shell script in the `/etc/init.options` file using the `-sc` option:

  ```
  -sc /etc/rc       shell script = /etc/rc
  ```

The file /etc/rc is the default z/OS UNIX initialization shell script that is executed when z/OS UNIX is started. See the *z/OS Communications Server: IP Migration*. Information such as the following can be entered in /etc/rc:

```
# Start name server
/usr/lpp/tcpip/sbin/named -b /named/production/named.boot &
```

Port 53 must be reserved for NAMED in the PROFILE.TCPIP data set. For directions on specifying port ownership, see "Step 3. Specify Port Ownership" on page 335. When the stack to which named binds is started, named completes initialization.

- Use the AUTOLOG statement to start the name server automatically during initialization with z/OS UNIX running. Insert the name of the named start procedure in the AUTOLOG statement of the PROFILE.TCPIP data set.

  ```
  AUTOLOG
    NAMED
  ENDAUTOLOG
  ```

  The job name keyword should not be added to the AUTOLOG statement for named. For more information on the AUTOLOG statement, refer to *z/OS Communications Server: IP Configuration Reference*.

  Note: Named cannot be started from INETD.

Step 12. Verify that the Name Server Started Correctly

After starting the name server, ensure that no errors occurred when it was started. Look in the syslog daemon output data set for name server messages. If start up was successful, something similar to the following messages are displayed:

```
named[22]: EZZ6698I name server starting. @(#) ddns/ns/ns_main.c,
           dns_ns, dns_r1.1 1.62  9/23/97 10:57:21
named[22]: EZZ6701I named established affinity with 'TCPCS'
named[22]: EZZ6540I Static primary zone 'mycorp.com' loaded (serial 1)
named[22]: EZZ6540I Static primary zone '34.37.9.in-addr.arpa' loaded (serial 1)
named[22]: EZZ6540I Static primary zone '0.0.127.in-addr.arpa' loaded (serial 1)
named[22]: EZZ6540I Static cache zone '' loaded (serial 0)
named[23]: EZZ6475I named:  ready to answer queries.
```

To correct errors, either stop and restart the name server to pick up the changes, or reload the name server with the -SIGHUP signal.

To stop the name server from the z/OS UNIX shell, issue:

```
kill -9 $(cat /etc/named.pid)
```

To stop the name server from the MVS console, issue the following:

```
p named1
(Use the name of the procedure that is currently active. This is
usually the proc name that was used to start the name server, followed
by a '1')
```

To reload the name server with a signal, issue the following command from the z/OS UNIX shell:

```
kill -HUP $(cat /etc/named.pid)
```

After restarting or reloading the name server, check again for errors.

Step 13. Verify the Name Server Can Accept Queries

When the name server is up with no logged errors, ensure that it can accept queries. Ensure that the name server can accept queries locally from both the MVS

and z/OS UNIX environments. In order to correctly set up these environments, see “Understanding Search Orders of Configuration Information” on page 6 for instructions.

After the resolver configuration is correct, test with the nslookup command.

Issue the following command from both the z/OS UNIX shell and the TSO ready prompt. In the following example, the name ’elmer.mycorp.com’ is used for the search.

Note: Choose any name in the domain you have defined.

```
nslookup elmer.mycorp.com
```

Using the sample files in this example, the following should be the result when the command is issued:

```
$ nslookup elmer.mycorp.com
Server:  localhost
Address:  127.0.0.1

Name:    elmer.mycorp.com
Address:  9.37.34.7
```

Note: The result should be the same in both environments.

Configuring a Secondary Name Server

After setting up a working primary name server, one or more secondary name servers can be set up. This process is very similar as configuring a primary name server. The differences are in the boot file and the domain data files.

For example, see “Configuring a Primary Name Server” on page 332 to configure a secondary server for the forward and reverse mapping zones. The steps are identical to the steps for configuring a primary name server, except where indicated below:

1. Create the boot file.
2. Specify stack affinity (multiple stack environment).
3. Specify port ownership.
4. Update the name server start procedure.
5. Create the domain data files (primary name server only).
6. Create the hints (root server) file.
7. Create the loopback file.
8. Ensure that the syslog daemon is running on your system.
9. Specify whether the name server is to run swappable or nonswappable.
10. Start the name server.
11. Verify that the name server started correctly.
12. Verify that the name server can accept queries.

The difference between the configuring a primary and secondary name server is the creation of domain data files (the database files containing host-to-address and address-to-host mappings). The domain data files are maintained on the primary name server, and the secondary name server transfers this data to its own database.

Instructions for creating the boot file for a secondary name server are below. All remaining steps are identical to those in “Configuring a Primary Name Server” on page 332.

Step 1. Create the Boot File

The easiest way to create a boot file for a secondary name server is to start from the boot file for the primary name server. The sample boot file, SEZASAMP(NAMED), modified to reflect the setup for a secondary name server is shown below.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;             5647-A01 (C) COPYRIGHT IBM CORP. 1997
;
; COMPONENT_NAME: TCPIP named.boot
;
; FUNCTIONS:
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUP
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;
;  /etc/named.boot
;
;        boot file for name server
;
;
; type      domain                source file or host
;
directory   /etc/dnsdata
secondary   mycorp.com            9.37.34.149 named.for
secondary   34.37.9.in-addr.arpa  9.37.34.149 named.rev
primary     0.0.127.in-addr.arpa  named.lbk
cache       .                     named.ca
options     query-log
```

This boot file specifies:

1. The location of the files (/etc/dnsdata)
2. That the name server will be the secondary name server for the `mycorp.com` zone
3. That the primary name server is located at IP address `9.37.34.149`
4. That the data for `mycorp.com` will be stored in /etc/dnsdata/named.for after it is retrieved from the primary name server by doing a zone transfer
5. That the name server will be the secondary name server for the reverse mapping zone, `34.37.9.in-addr.arpa`

6. That the primary name server is located at IP address 9.37.34.149
7. That the data for addresses 9.37.34.x contained in the zone will be stored in /etc/dnsdata/named.rev after it is retrieved from the primary name server by doing a zone transfer
8. That the name server will continue to function in the primary role for the loopback address (127.0.0.1)
9. That the list of root name servers is in /etc/dnsdata/named.ca
10. That all queries coming in to this name server will be logged in the syslog daemon output file

Configuring a Cache-only Name Server

If the name server does not need to be authoritative for any data, choose a special type of name server, called a caching-only name server. A caching-only name server can improve performance by reducing the number of network flows required for names or addresses that are frequently requested.

Follow these steps to configuring a basic name server:

1. Create the boot file.
2. Specify stack affinity (multiple stack environment).
3. Specify port ownership.
4. Update the name server start procedure.
5. Create the domain data files (primary name server only).
6. Create the hints (root server) file.
7. Create the loopback file.
8. Ensure that the syslog daemon is running on your system.
9. Specify whether the name server is to run swappable or nonswappable.
10. Start the name server.
11. Verify that the name server started correctly.
12. Verify that the name server can accept queries.

The difference between configuring a primary name server and configuring a caching-only server is the creation of domain data files (the database files containing host-to-address and address-to-host mappings).

Step 1. Create the Boot File

The easiest way to create a boot file for a caching-only name server is to use the boot file for the primary name server. The sample boot file /usr/lpp/tcpip/samples/named.boot, modified to reflect the setup for a caching-only name server, is shown below.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;             5647-A01 (C) COPYRIGHT IBM CORP. 1997
;
; COMPONENT_NAME: TCPIP named.boot
;
; FUNCTIONS:
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
```

```
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUP
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;
;  /etc/named.boot
;
;        boot file for name server
;
;
; type         domain                   source file or host
;
directory     /etc/dnsdata
primary       0.0.127.in-addr.arpa     named.lbk
cache         .                        named.ca
options      query-log
```

This boot file specifies:

1. The location of the files (/etc/dnsdata)
2. That the name server will continue to function in the primary role for the loopback address (127.0.0.1)
3. That the list of root name servers is in /etc/dnsdata/named.ca
4. All queries coming in to this name server will be logged in the syslog daemon output file

Adding Forwarding to Your Name Server

In order to use forwarding in any name server, update the boot file. Add the following statement to the boot file:

```
forwarders 9.4.2.1
```

where 9.4.2.1 is the IP address of the machine where queries should be forwarded. This sends unresolved queries to 9.4.2.1 before trying to resolve the query using root name servers (specified in the hints file) or other cached name servers authoritative for or 'closer' to the authoritative name server.

For a name server to *only* use forwarders and not use the root servers, in addition to the forwarders directive, also add the following directive to its boot file:

```
options forward-only
```

A name server with this option can still answer queries from its cached data. The cache is checked first and if the cache does not contain the answer, the query is sent to the name server(s) in the forwarders list.

Configuring Host Resolvers: Name Server Considerations

If the name server will run on the host being configured, create a loopback file. Specify the loopback address in the *first* name server directive of the resolver configuration file so local clients can access the name server. Refer to "Step 7. Create the Loopback File" on page 340 for loopback address considerations.

The name server uses a private resolver that is different from the LE resolver used by other z/OS UNIX socket programs. The name server has the following functional differences:

- onslookup does not use site tables (for example, /etc/hosts) for host name resolution.
- Only the built-in translation table is used.

For a complete discussion of resolver configuration files, see *z/OS Communications Server: IP Configuration Reference*.

Creating the Syslog File

If your syslog daemon is not configured, see "Configuring the Syslog Daemon (syslogd)" on page 63 for information regarding the syslog daemon.

Syslog daemon (syslogd) is a server process that is typically started as one of the first processes in an z/OS UNIX environment. Servers and stack components use syslogd for logging purposes and can also send trace information to syslogd. The named daemon logs messages to the syslog daemon. For information about the syslog daemon, see "Configuring the Syslog Daemon (syslogd)" on page 63.

The name and location of your syslog file is specified in `/etc/syslog.conf`.

Special Considerations When Using Dynamic VIPA

If you run a name server on a host that is using Dynamic VIPA (DVIPA), you may be required to do some additional configuration. Name servers running on a host using DVIPA need to BIND the UDP port that the name server listens on (usually 53) to the DVIPA, if you wish DNS to make use of the DVIPA. This can be done by using the BIND option on the UDP PORT statement in the TCPIP.PROFILE. If you do BIND the DNS UDP port to the DVIPA, then all references to that name server must use the DVIPA whether those references are from other name servers or from resolvers.

References to a name server could occur in a number of places, and should be changed to use the DVIPA if BINDing a DNS UPD port to the DVIPA. This list is not exhaustive, but is intended to aid you for some of the most common cases. In general, you may need to change any place that references a name server by its IP address when using DVIPAs.

Task	Location
Delegating a DNS subdomain to a name server running on a host using DVIPA	The 'A' record in the glue records for the delegated (child) name server in the domain data file of the delegating (parent) name server
Designating a secondary (slave) name server when the primary (master) name server is running on a host using DVIPA	The 'secondary' statement in the boot file of the secondary (slave) name server
Configuring resolvers	'NSINTERADDR' or 'nameserver' directive of the resolver configuration file
Using a name server as the target of other forwarding name servers when the target name server resides on a host using DVIPA	'forwarders' directive in the boot file of the forwarding name servers
Using a name server as an intranet root name server when the root name server is running on a host using DVIPA	'A' record of the intranet root name server in the hints file on all name servers within the intranet

Querying Name Servers

This section describes how to use the onslookup command to query the name server.

Notes:

1. The onslookup command runs only from the z/OS shell. The nslookup command can query the name server from TSO or the z/OS shell.
2. The host command is another way to query name servers from the z/OS shell. See the *z/OS Communications Server: IP User's Guide* for a list of host commands.

onslookup/nslookup Command

The onslookup and nslookup command can be used to query the name server to perform the following tasks:

- Identifying the location of name servers
- Examining the contents of a name server database
- Establishing the accessibility of name servers

The onslookup command has two modes of operation: interactive mode and command mode. In either mode, the address of the default name server comes from the resolver configuration data. In the sample data below, the default domain is raleigh.ibm.com, and the default name server is at 9.37.34.149. If that name server fails to respond, the one at 9.37.34.7 is used.

```
domain   raleigh.ibm.com
nameserver   9.37.34.149
nameserver   9.37.34.7
```

Entering the Interactive Mode

Interactive mode can be used to repetitively query one or more name servers for information about various hosts and domains, to display that information on the console, and, in some cases, to write response data to a file.

You can enter the interactive mode under the following conditions only:

- No arguments are supplied on command invocation; the default name server is used.
- The first argument is a hyphen, and the second argument is the host name or Internet address of a name server.

For a complete description of the onslookup and nslookup interactive modes, refer to *z/OS Communications Server: IP User's Guide*.

Entering the Command Line Mode

The command line mode displays or stores the output from the query supplied as part of the invocation string and then exits.

To enter the command line mode, provide a complete query with the onslookup command invocation string.

For a complete description of the onslookup and nslookup command line modes, refer to *z/OS Communications Server: IP User's Guide*.

onslookup/nslookup Configuration

The configuration options of onslookup determine the operation and results of the name server queries. The values for onslookup options can be specified in more than one location, as shown in Table 15. Values specified as onslookup command options have priority over values specified in the `.onslookuprc` file, which have priority over the value specified by the environment variable, and so on. For example, the value specified by the `all` option in the onslookup command has priority over the value specified by the `all` option in the `.onslookuprc` file. Similarly, the value specified by `ResolverTimeout` in the `/etc/resolv.conf` file has priority over the value specified by `ResolverTimeout` in the TCPIP.DATA configuration data set.

The letters beside some settings indicate that the *terms* are functionally equivalent. For example, the term `domain` (the letter ″A″) is functionally equivalent to the terms `DomainOrigin` and `LOCALDOMAIN`. If two functionally equivalent settings are listed in the same file, the one listed last has priority. For example, if `domain` and `DomainOrigin` are both listed in the `/etc/resolv.conf` file and `domain` is listed first, the value specified by `DomainOrigin` has priority.

Note: If you are using `/etc/resolv.conf` the file must contain all the data normally found in the TCPIP.DATA data set. If information is read from `/etc/resolv.conf`, then TCPIP.DATA is not used. To configure the `/etc/resolv.conf` file, you use certain directives.

The numeric column headings in Table 15correspond to the following:

1 onslookup command options. Refer to *z/OS Communications Server: IP User's Guide* for more details about the onslookup command.

2 .onslookuprc file in the home directory. Refer to *z/OS Communications Server: IP User's Guide* for more details.

3 Environment variable (set by the USS command ″export LOCALDOMAIN=domain_origin″).

4 `/etc/resolv.conf`

5 TCPIP.DATA configuration data set. Refer to *z/OS Communications Server: IP Configuration Reference* for more details.

For example, column 1 lists onslookup options and column 2 lists options you can set in the `.onslookuprc` file.

Table 15. Settings That Affect onslookup/nslookup Operation

Settings	1	2	3	4	5
All	x	x			
Class	x	x			
no[d2]	x	x			
[no]debug	x	x			
[no]defname	x	x			
domain (A)	x	x		x	
[no]ignoretc	x	x			
port (B)	x	x			
querytype	x	x			
[no]recurse	x	x			

Table 15. Settings That Affect onslookup/nslookup Operation (continued)

Settings	1	2	3	4	5
retry (C)	x	x			
root	x	x			
[no]search	x	x			
srchlist (D)	x	x			
timeout (E)	x	x			
[no]vc (F)	x	x			
search (D)				x	
nameserver (G)				x	
sortlist				x	
options debug				x	
options ndots				x	
DomainOrigin (A)				x	x
NsInterAdd (G)				x	x
NsPortAddr (B)				x	x
ResolveVia (F)				x	x
ResolverTimeout (E)				x	x
ResolverUdpRetries (C)				x	x
LOCALDOMAIN (A)			x		

Resolver configuration directives for `/etc/resolv.conf` are listed below.

domain *domain name*
Specifies the resolver's default domain if the host name is not fully qualified.

search *domain domain...*
Specifies an ordered list of domains for the resolver to search. The domain listed first is searched first.

nameserver *IP_address(es)*
Specifies the IP address or addresses of a particular name server to query. The addresses are queried in the order listed.

sortlist *subnet {/subnet_mask}...*
Specifies an ordered list of subnets and networks if the resolver receives more than one address as a result of a query. You can include one or more subnet masks or you can omit the mask to specify the entire network.

options *debug*
Starts the resolver debugging option.

options ndots: *minimum_number_of_dots*
Specifies the minimum number of dots an argument must have before the search list is applied. If the argument has less than the specified number of dots, the search list is appended to the name before any queries are sent. If the argument has the same number of dots or more, the query is sent first just as the user typed it. If a positive response is not received, subsequent queries are sent with the search list appended. The default is 1.

Programs that query a name server are called resolvers. Because many TCP/IP applications need to query the name server, a set of routines is usually provided for

application programmers to perform queries. Under MVS, these routines are available in the TCP/IP application programming interface (API) for each supported language or LE for z/OS UNIX Sockets API.

The onslookup command uses a private resolver that is different from the LE resolver used by other z/OS UNIX socket programs. The onslookup command has the following functional differences:

- The HFS file, `/etc/hosts`, is required for host table lookup if name services do not exist. Following is a sample /etc/hosts file:
- The search order for onslookup's resolver configuration file differs from other resolvers. The search order is:
  1. The file specified by the environment variable, RESOLVER_CONFIG
  2. /etc/resolv.conf
  3. File defined by the //SYSTCPD DD statement in the user's TSO logon procedure
  4. userid.TCPIP.DATA
  5. SYS1.TCPPARMS(TCPDATA)
- onslookup does not use SiteTables (for example, /etc/hosts) for host name resolution.
- Only the built-in translation table is used.
- The 'search' directive is only supported by the nslookup resolver.
- The 'sortlist' directive is only supported by the nslookup resolver.
- The 'options' directive is only supported by the nslookup resolver.
- The 'options ndots:' directive is only supported by the nslookup resolver.
- onslookup's private resolver uses the LOCALDOMAIN environment variable, whereas no other resolvers use this.

For a complete discussion of resolver configuration files, see "Chapter 1. Configuration Overview" on page 3.

If the name server will run on the host being configured, you need to configure the *first* name server (or NsInterAddr) directive in the resolver configuration file as the loopback address (127.0.0.1 or any address in your home list).If any VIPA addresses are used with the NsInterAddr statement, ensure that IPCONFIG SOURCEVIPA is coded in PROFILE.TCPIP. If it is not, UDP packets returned from the VIPA address will have the physical interface address as the destination address instead of the VIPA address that it sent. The UDP packet will be discarded when it is received because the addresses do not match.

Diagnosing Problems

This section describes four methods for diagnosing problems:

- Checking messages on the operators console.
- Checking the syslog messages.
- Using name server signals.
- Using the onslookup program.

These methods are discussed below. In addition to these methods, diagnosing problems for a dynamic zone can be done with nsupdate.

Checking Messages Sent to the Operators Console

Messages displayed on the operators console indicate the status of your DNS. Messages fall into the following categories:

- Name server initialization
- Name server initialization failure
- Name server initialization complete (always EZZ6475I)
- Name server termination

Regularly check console messages to identify problems.

Checking the Syslog Messages

Error messages are also displayed in the syslog output file, which is pointed to by the syslog configuration file. (`/etc/syslog.conf` is the default configuration file.) For descriptions of the syslog file and the syslog daemon, see "Configuring the Syslog Daemon (syslogd)" on page 63.

Using Name Server Signals to Diagnose Problems

You can use name server signals to send messages to the named daemon. These signals control various functions that can be used to diagnose problems.

Diagnostic functions include the following:

- Enabling and disabling debug message logging
- Dumping the contents of the name server database
- Getting short status
- Logging queries

For an explanation of possible output messages, refer to publications like *DNS and Bind* by Albitz and Liu.

Using onslookup/nslookup to Diagnose Problems

The onslookup program lets you query other name servers with the same query packet another name server would use. This is helpful in diagnosing lookup problems in TCP/IP.

It is recommended that you use onslookup or nslookup with each NsInterAddr used in TCPIP.DATA to ensure you receive the expected results. Some name server clients, on other platforms, may require the address you specify for the name server to match the source IP address in the response from the name server. For example, if a static VIPA address is specified as the address of the name server, and IPCONFIG SOURCEVIPA is not specified in PROFILE.TCPIP, then nslookup on some platforms will discard the returned packet because it will have the destination address of the physical interface instead of the VIPA interface. If you wish to specify a Dynamic VIPA (DVIPA) as the address of the name server, then the name server must BIND the UDP port to the DVIPA. Refer to *z/OS Communications Server: IP Configuration Reference* for information on how to specify the BIND parameter on the PORT statement in the TCPIP.PROFILE.

To turn debugging on at level 1, enter the following commands from the z/OS shell:

```
onslookup
set debug
```

The onslookup program shows timeouts and displays response packets. To turn the debug option off, enter the following command:

```
set nodebug
```

You can set the debugging option to level 2 by entering the following commands:

```
onslookup
set d2
```

The resolver shows the normal debugging information plus the query packets that were sent out. Turning on `d2` also turns on `debug`. Turning off `d2`, however, only turns off `d2` and `debug` remains on. To turn off both `d2` and `debug`, turn off `debug` by entering the subcommand and option `set nodebug`.

Sample Files

The following sections provide sample files. The notes after each sample explain some of the highlights.

Sample Forward Domain Data File

Following is the sample forward domain data file shipped as /usr/lpp/tcpip/samples/named.for. Refer to the program directory for its location. The file specifies the host systems in three networks (7.0.0.0, 8.0.0.0, and 9.0.0.0). The host system dog is a router between two of the networks.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;             5647-A01 (C) COPYRIGHT IBM CORP. 1997, 2000
;
; COMPONENT_NAME: TCPIP named.data
;
; FUNCTIONS: nameserver
;
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1989, 2000
; All Rights Reserved
; Licensed Materials - Property of IBM
;
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;

;   /etc/named.for file
     ;
     $ORIGIN ibm.com.
     raleigh    IN  SOA  buzz1.raleigh.ibm.com. bug@vmail (                  {1}
        1       ; Serial--  change when database is changed
        10800   ; Refresh-- secondary checks every three hours
        3600    ; Retry--   secondary retries connection
```

```
            ;            every hour after a failed zone transfer
      604800  ; Expire--  data expires after one week
      86400 ) ; Time to Live-- data cached in other servers one day)
    ;
    $ORIGIN raleigh.ibm.com.                                          {2}
                IN      NS      buzz1                                 {3}
                IN      NS      elmer
                IN      NS      jumbo
    _http._tcp          SRV     0  0 80   buzz1.ibm.com.             {4}
                        SRV     10 0 8000 elmer.ibm.com.              {4}
    _http._tcp.www      SRV     0  0 80   buzz1.ibm.com.             {5}
                        SRV     10 0 8000 elmer.ibm.com.              {5}
    ; OWNER     CLASS   TYPE      RECORD DATA
    localhost    IN      A       127.0.0.1
    buzz1        IN      A       9.37.34.149
    elmer        IN      A       9.37.34.7
    jumbo        IN      A       8.37.34.12
    dog          IN      A       8.37.34.113                          {6}
                 IN      A       9.37.34.113
    vmail        IN      A       7.37.34.1
    kent         IN      A       7.37.34.2
    greg         IN      A       7.37.34.3
    printserver  IN      A       7.37.34.4
                 IN      A       8.37.34.33
                 IN      A       9.37.34.44
    gary         IN      CNAME   jumbo                                {7}
```

{1} The SOA (Start of Authority) record specifies the name server buzz1 as the authoritative name server for the domain raleigh.ibm.com. The mail address of the person responsible for domain data is bug@vmail. The numbers enclosed in parentheses are parameters used primarily to manage the zone information.

{2} The control entry $ORIGIN appends the string raleigh.ibm.com. to all the following host names that do not end with a dot (.).

{3} The NS (Name Server) records specify the name servers in the zone. Note that NS records do not distinguish between primary and secondary name servers.

{4} This A (Address) record maps the host name (dog.raleigh.ibm.com) to the IP addresses of the two networks to which it is connected.

{5} The CNAME record specifies that the name gary is an alias for the host name jumbo.raleigh.ibm.com.

{6} This A (Address) record maps the host name (dog.raleigh.ibm.com) to the IP addresses of the two networks to which it is connected.

{7} The CNAME record specifies that the name gary is an alias for the host name jumbo.raleigh.ibm.com.

Sample Reverse Domain Data File

The following is a sample reverse domain data file that maps IP addresses to host names in the network 9.37.0.0. The addresses of host systems in the other two networks (7.0.0.0 and 8.0.0.0) would be listed in separate reverse domain data files.

```
;    /etc/named.rev

$ORIGIN 37.9.in-addr.arpa.
34 IN  SOA  buzz1.raleigh.ibm.com. bug@raleigh.ibm.com. (
               1  10800  3600  604800  86400 )          {1}
               IN      NS      buzz1.raleigh.ibm.com.   {2}
               IN      NS      elmer.raleigh.ibm.com.
```

```
$ORIGIN 34.37.9.in-addr.arpa.                              {3}
7               IN      PTR     elmer.raleigh.ibm.com.    {4}
149             IN      PTR     buzz1.raleigh.ibm.com.
113             IN      PTR     dog.raleigh.ibm.com.
44              IN      PTR     printserver.raleigh.ibm.com.
```

{1} These lines are the start of authority (SOA) entry, which lists information about the domain, including the authoritative name server. Like a forward domain data file, a reverse domain data file can have only one SOA resource record. The left and right parentheses mark the beginning and end, respectively, of the parameters used to manage zone information.

{2} These NS (Name Server) records specify two name servers, buzz1 and elmer, as the authoritative name servers in the zone.

{3} The $ORIGIN control entry appends the string 34.37.9.in-addr.arpa. to all addresses not ending with a dot.

{4} This pointer (PTR) entry maps the IP address 7.34.37.9.in-addr.arpa. to the host name elmer.raleigh.ibm.com.

Sample Hints (Root Server) File

The hints file contains the names and addresses of name servers in the root domain.

```
;   This file holds the information on root name servers needed to
;   initialize cache of Internet domain name servers
;   (e.g., reference this file in the "cache <file>"
;   configuration file of BIND domain name servers).
;
;   This file is made available by InterNIC registration services
;   under anonymous FTP as
;   file                /domain/named.root
;   on server           FTP.RS.INTERNIC.NET
;   -OR- under Gopher** at RS.INTERNIC.NET
;       under menu      InterNIC Registration Services (NSI)
;         submenu       InterNIC Registration Archives
;       file            named.root
;
;   last update:       May 19, 1997
;   related version of root zone:      1997051700
;
;   formerly NS.INTERNIC.NET
;
.                     3600000   IN  NS   A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.   3600000       A    198.41.0.4
;
;   formerly NS1.ISI.EDU
;
.                     3600000       NS   B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.   3600000       A    128.9.0.107
;
;   formerly C.PSI.NET
;
.                     3600000       NS   C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.   3600000       A    192.33.4.12
;
;   formerly TERP.UMD.EDU
;
.                     3600000       NS   D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.   3600000       A    128.8.10.90
;
;   formerly NS.NASA.GOV
;
.                     3600000       NS   E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.   3600000       A    192.203.230.10
```

```
;
;  formerly. NS.ISC.ORG
;
.                       3600000       NS   F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.   3600000       A    192.5.5.241
;
;   formerly NS.NIC.DDN.MIL
;
.                       3600000       NS   G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.   3600000       A    192.112.36.4
;
;   formerly AOS.ARL.ARMY.MIL
;
.                       3600000       NS   H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.   3600000       A    128.63.2.53
;
;   formerly NIC.NORDU.NET
;
.                       3600000       NS   I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.   3600000       A    192.36.148.17
;
;   temporarily housed at NSI (InterNIC)
;
.                       3600000       NS   J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.   3600000       A    198.41.0.10
;
;   housed in LINX, operated by RIPE NCC
;
.                       3600000       NS   K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.   3600000       A    193.0.14.129
;
;   temporarily housed at ISI (IANA)
;
.                       3600000       NS   L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.   3600000       A    198.32.64.12
;
;   temporarily housed at ISI (IANA)
;
.                       3600000       NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.   3600000       A    198.32.65.12
; End of File
```

Sample Loopback File

Loopback addresses let hosts and clients direct communications to themselves. The following is a sample loopback file used for all three name servers in the raleigh.ibm.com domain (buzz1, elmer, and jumbo).

```
;    /etc/named.lbk

0.0.127.in-addr.arpa. IN SOA   buzz1.raleigh.ibm.com. bug@vmail (   {1}
     1
     10800
     3600
     604800
     86400    )
0.0.127.in-addr.arpa.     IN  NS   buzz1.raleigh.ibm.com.    {2}
0.0.127.in-addr.arpa.     IN  NS   elmer.raleigh.ibm.com.
0.0.127.in-addr.arpa.     IN  NS   jumbo.raleigh.ibm.com.

1.0.0.127.in-addr.arpa.   IN PTR   localhost.                {3}
```

{1} These records are the SOA (Start of Authority) entry, which lists information about the zone. The left and right parentheses mark the beginning and end, respectively, of the parameters used to manage zone information.

{2} These three records identify the three name servers in the zone.

{3} The PTR (Pointer) record indicates the common loopback address of the host systems.

Sample Boot File

A boot file directs a name server to its data files. The sample boot file below is provide with the product. Refer to the program directory for its location.

```
;   /etc/named.boot
;
; type           domain              source file or host
directory     /etc/dnsdata                          {1}
primary       raleigh.ibm.com           named.for   {2}
primary       34.37.9.in-addr.arpa      named.rev   {3}
primary       0.0.127.in-addr.arpa      named.lbk   {4}
cache      .                            named.ca    {5}
options query-log                                   {6}
```

{1} The directory directive specifies `/etc/dnsdata` as the working directory for the boot file and eliminates the need for a file path for the data files, loopback file, and hints file.

{2} This primary directive specifies the server on which this boot file is installed as the primary server for the domain raleigh.ibm.com. The directive also specifies that the name-to-address mappings are in the forward domain data file, `named.for`.

{3} This primary directive specifies the server on which this boot file is installed as the primary server for the reverse domain 34.37.9.in-addr.arpa and specifies that the address-to-name mappings are in the reverse domain data file, `named.rev`.

{4} This primary directive specifies the loopback network and file for the host.

{5} The cache directive specifies the hints file, which lists the top-level domain name servers. The file path and name of the hints file is `/etc/named.ca`.

{6} This directive tells the name server to log all the queries it receives to the syslog log file.

The following is an example of a caching-only name server boot file:

```
directory /usr/local/named   ; or your data directory
primary 0.0.127.in-addr.arpa  db.127.0.0    ; for loopback address
cache   .                     db.cache
```

Advanced Name Server Topics

Connection Optimization in a Sysplex Domain

This section describes *connection optimization*, a technique that uses DNS for balancing IP connections and workload in a sysplex domain. It also dynamically manages the active and available list of IP addresses for resources in a sysplex domain.

Overview

Connection optimization uses DNS for distributing connections among hosts or server applications within a sysplex domain. A sysplex is a set of MVS systems communicating and cooperating with each other through multisystem hardware and software components.

In DNS terms, a sysplex is a subdomain that is added to the DNS name space. Name servers running within the sysplex perform name resolution. Resolvers query these name servers directly or indirectly through the name server authoritative for the resources in the sysplex domain.

Connection optimization extends the concept of a ″DNS host name″ to clusters, or groups of server applications or hosts. Server applications within the same group are considered to provide equivalent service. Connection optimization utilizes round-robin logic and load-based ordering to determine which addresses to return for a given cluster.

Connection optimization increases overall efficiency by favoring connections to systems with the most available resources and by avoiding unavailable sysplex resources. Addresses of the most available server applications or hosts are returned more frequently than the addresses of loaded server applications or hosts.

A connection-optimized sysplex domain is also scalable—that is, it can add servers and interface addresses dynamically to provide more service capacity. Client applications have dynamic access to the addresses of those servers, with no DNS restart or administration required.

Registration: To ensure maximum availability, server applications register with Workload Manager (WLM), which quantifies the availability of server resources within a sysplex. WLM must be configured in goal mode on all hosts within the sysplex. See “Step 7: Configure WLM in Goal Mode” on page 371 for a description of this procedure.

TCP/IP stacks also register with WLM. Additionally, they provide the active IP addresses. For a description of how IP addresses are associated with a sysplex domain name, see “Associating IP Addresses with the Sysplex Domain Name” on page 361. For a discussion of TCP/IP configuration issues, see “Configuring TCP/IP” on page 367.

When registering, server applications provide the following information:

- *Group name*. This is the name of a cluster of equivalent server applications in a sysplex. It is also the name within the sysplex domain that client applications use to access the server applications. To connect to *any* server application in a group, a client application uses the combination *group_name.sysplex_domain_name*.

  Note: The group name ″TCPIP″ and the group name for the sysplex domain (for example, ″mvsplex″) are reserved and cannot be used by server applications.
- *Server name*. This is the name of the server application instance. The server name must be unique among all servers that share the same group name. A server application instance can belong to more than one group.
- *Host name*. This is the host name of the TCP/IP stack on which the server application runs.

The sysplex domain name should be registered with the domain name server under a special address, 127.0.0.128, which is used by the ioctl() SIOCGSPLXFQDN. For details, see “Configuring a Primary Name Server” on page 332.

Name Resolution: In connection optimization, a name server performs resolution for a name representing a cluster of hosts or server applications. Figure 48 on page 360

page 360 depicts a sysplex domain called mvsplex.mycorp.com. The sysplex domain contains only the resources participating in the sysplex. Client applications append the domain name, mycorp.com, to all requests for name resolution.

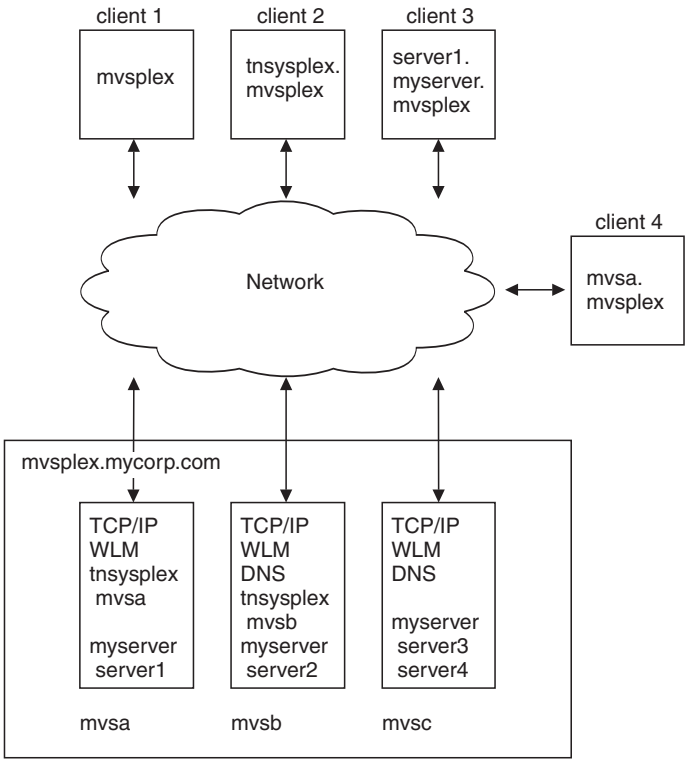


Figure 48. Name Resolution to a Sysplex

Each host system runs TCP/IP. Hosts mvsb and mvsc are also running the z/OS UNIX sysplex name servers (DNS). mvsb is running the primary name server for the sysplex subdomain, and mvsc is running the secondary.

Each host is running one or more myserver server applications, and mvsa and mvsb are also running tnsysplex server applications. The group names (tnsysplex and myserver), the tnsysplex server names (mvsa and mvsb), and the myserver server names (for example, server1) are known to the name server through WLM registration by the respective application.

Note: tnsysplex represents a cluster of TN3270 server applications. Every instance of TN3270 running on a particular host registers using the same server name.

Four types of requests are shown in Figure 48. Client Application 1 requests the services of any host on the system by providing only the name of the sysplex subdomain, mvsplex, to DNS for resolution. Client Application 2 requests the services of any tnsysplex server instance running in the sysplex. Client Application 3 requests a particular server instance in the myserver group, and Client Application 4 requests connection to a particular host.

In Figure 48, only Client Applications 1 and 2 are candidates for connection optimization. Client Application 1 can connect to either mvsa, mvsb, or mvsc. Client Application 2 can connect to either mvsa or mvsb (but not mvsc). Client Applications 3 and 4, which can connect only to mvsa, are ineligible for connection optimization.

They do, however, benefit from the round-robin selection process DNS uses to balance across available network interfaces.

Note: When using connection balancing via MVS clients (such as FTP), to avoid caching of IP addresses, during name resolution set the following environment variable:

```
export _EDC_IP_CACHE_ENTRIES=0
```

Generated Names vs. Statically Defined Names: All name servers use *statically defined* names. These are the names in the forward domain data file. As the DNS administrator for a name server using connection optimization, statically define the names of the hosts in the sysplex and the NS and SOA resource records in the forward domain data file. For a description of sysplex forward domain data files, see "Sysplex Data Files" on page 369.

Note: The host names in the data files must match the host names specified in the stack's TCPIP.DATA data set and should be 20 characters or less to ensure server uniqueness.

A name server using connection optimization also uses *generated names*. These are added dynamically to the domain name space as TCP/IP stacks and server applications in the sysplex register with WLM. The name server uses three types of generated names. In Figure 48 on page 360, the following generated names are used:

- The group name that is registered by the server applications (`tnsysplex` and `myserver`)
- The server name concatenated with the group name of each server application that registers with WLM in the sysplex (for example, `server1.myserver`)
- An alias for the sysplex domain name, `mvsplex`

The generated names become resources (or ″host″ names) within the sysplex domain, creating fully qualified domain names:

- Fully qualified group names (these are the connection balanced names):
  - `tnsysplex.mvsplex.mycorp.com`
  - `myserver.mvsplex.mycorp.com`
- Fully qualified server names:
  - `mvsa.tnsysplex.mvsplex.mycorp.com`
  - `server1.myserver.mvsplex.mycorp.com`
- Fully qualified alias name for the sysplex name `mvsplex.mycorp.com`:
  - `mvsplex.mvsplex.mycorp.com`

Associating IP Addresses with the Sysplex Domain Name: The sysplex domain name `mvsplex.mycorp.com` is associated with the *intersection* of the set of statically defined addresses associated with the stacks that are registered with WLM and the set of addresses associated with the adapters that are active on those stacks. The TCP/IP stack must be registered with WLM for this to occur. Figure 49 on page 362 depicts this intersection.

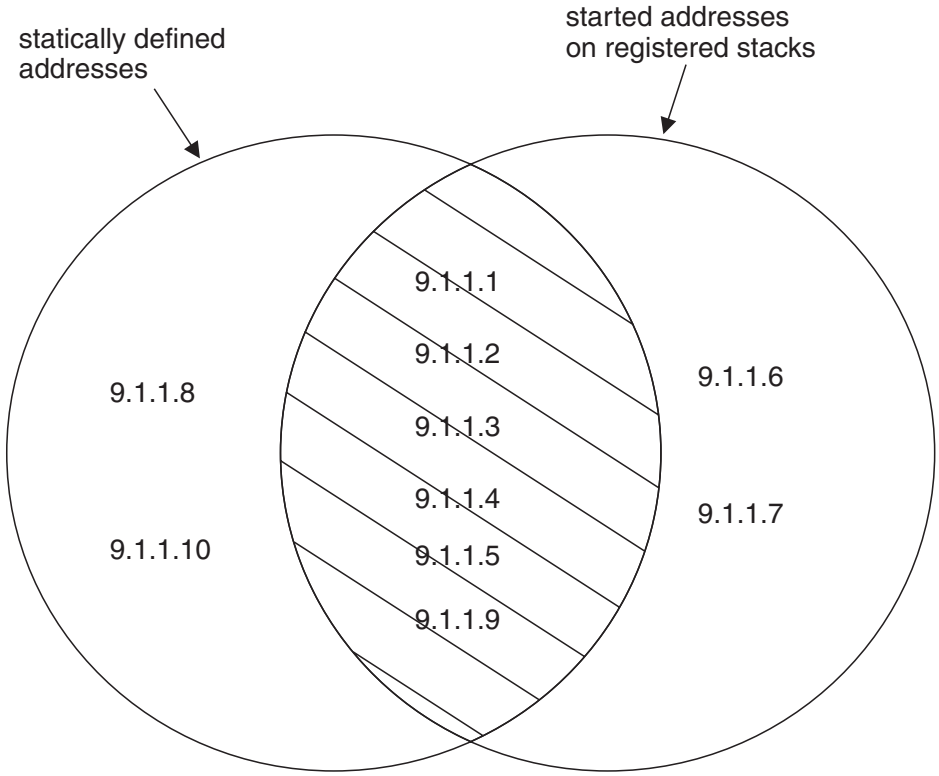


Figure 49. Address Association with mvsplex.mycorp.com

The hosts, `mvsa`, `mvsb`, and `mvsc` in the figure have the following addresses in the HOME statement in the PROFILE.TCPIP data set. Only certain adapters are active.

```
mvsa                     mvsb                      mvsc
9.1.1.1 (adapter active) 9.1.1.4 (adapter active)   9.1.1.8
9.1.1.2 (adapter active) 9.1.1.5 (adapter active)   9.1.1.9 (adapter active)
9.1.1.3 (adapter active) 9.1.1.6 (adapter active)   9.1.1.10
                         9.1.1.7 (adapter active)
```

The forward domain data file for the sysplex contains the following statically defined addresses:

```
mvsa            IN        A         9.1.1.1
                IN        A         9.1.1.2
                IN        A         9.1.1.3
mvsb            IN        A         9.1.1.4
                IN        A         9.1.1.5
mvsc            IN        A         9.1.1.8
                IN        A         9.1.1.9
                IN        A         9.1.1.10
```

Using the intersection of two sets lets the domain administrator selectively exclude certain IP addresses from use, such as 9.1.1.6 and 9.1.1.7 on mvsb, while distributing only active addresses to client applications. For instance, 9.1.1.8 would never be used since it is not active.

The process of associating IP addresses with server applications is similar to that for hosts. When a server application registers with WLM, the dynamically generated group name (for example, `myserver`) is added to the sysplex domain. If the stack on which the server application is running is registered with WLM, then the addresses associated with the group name are the intersection of the statically defined addresses for that stack and those addresses that are associated with adapters that

are active. When the server application is replicated on other hosts in the sysplex, the addresses associated with the group name are the union of all intersection sets.

Detailed Example: The following example describes in detail how IP addresses become associated with a server application `myserver` running in the sysplex `mvsplex.mycorp.com`. When reading the following section, refer to Figure 50.

The example is described in terms of a *process*, beginning initially with no registered servers applications or stacks. The statically defined addresses associated with the `mvs` hosts and coded in the forward domain data file are the statically defined addresses listed earlier. As server applications and stacks register, the IP addresses associated with `myserver.mvsplex.mycorp.com` change. Figure 50 shows the conclusion of this process.

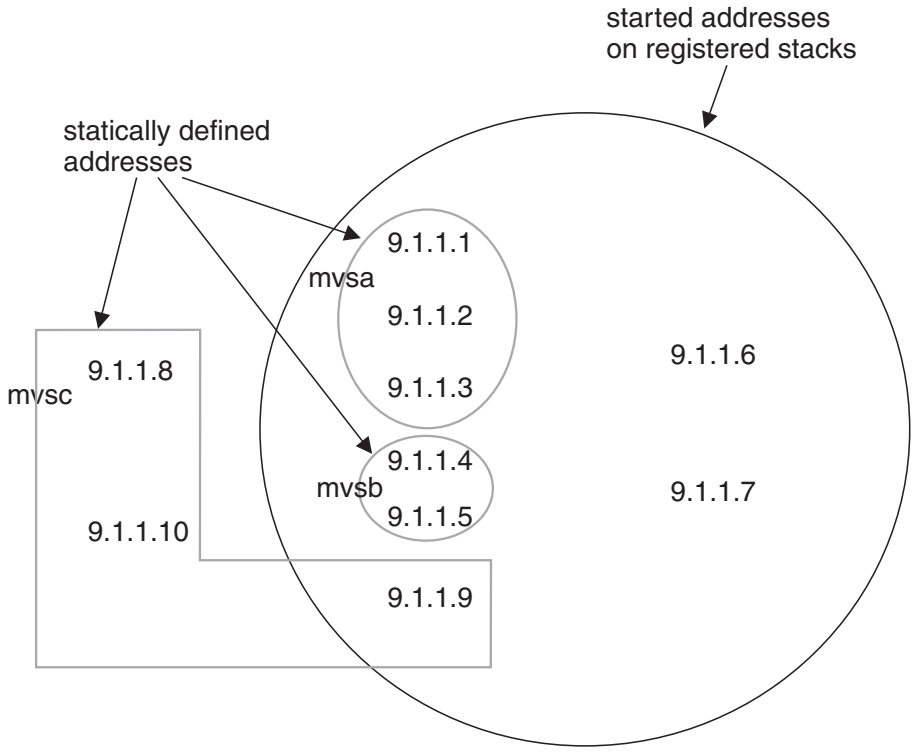


Figure 50. Address Association with myserver

Assume initially that a server application which registers with the group name `myserver` is running on `mvsc`, but that the stack on `mvsc` is not currently registered with WLM. The addresses associated with the name `myserver.mvsplex.mycorp.com` are `9.1.1.8`, `9.1.1.9`, and `9.1.1.10`.

Next, assume that the stack on `mvsc` registers with WLM. The addresses associated with the name `myserver.mvsplex.mycorp.com` are reduced to the intersection of the statically defined addresses for the stack and those addresses that are associated with adapters that are active. The only address associated with group name `myserver.mvsplex.mycorp.com` is thus `9.1.1.9`.

Now another instance of an equivalent server application registers with WLM on `mvsb`, but the stack on `mvsb` is *not* registered with WLM. All the statically defined addresses associated with `mvsb` are added to the set of addresses currently

associated with the group name myserver.mvsplex.mycorp.com (9.1.1.4, 9.1.1.5, and 9.1.1.9). If the addresses on mvsb are active, the set of addresses associated with the group name does not change.

Similarly, if another instance of an equivalent server application registers with WLM on mvsa and if the addresses on mvsa are active, the set of addresses associated with the server application myserver.mvsplex.mycorp.com are those shown in Figure 50 on page 363 (and Figure 49 on page 362).

Note: The adapters associated with addresses 9.1.1.6 and 9.1.1.7 are not associated with the dynamically generated group name because they are not statically defined in the forward domain data file.

Connecting to a Particular Server Instance: The server name with which the server application registers is unique among all other server instances that share the same group name. Client applications can use the server name to bypass connection optimization. For example, client applications can bypass connection optimization if they are in the middle of a transaction with a server application, and the client/server session fails before the transaction completes. If the client application software has the ability to recognize this situation, the client can reconnect to the same server instance and complete the transaction.

The group name is prefaced with the server name, separated by a dot (server_name.group_name). If the group name is myserver and if the application instance on mvsc registered with WLM as myserver3, then client applications can connect to that particular instance using the name myserver3.myserver.mvsplex.mycorp.com. The address associated with this name (9.1.1.9) is a subset of the addresses currently associated with the group name myserver.mvsplex.mycorp.com that exist on stack mvsc.

Usage Considerations in a Connection Optimized Sysplex: Connection optimization extends the concept of a ″DNS host name″ to include a name that generically represents (1) all hosts in the sysplex (provided their stacks register with WLM) and (2) names that represent groups of equivalent server applications spread across the sysplex (provided those server applications register with WLM). The maximum benefit from connection optimization is realized when all stacks in the sysplex register with WLM and all TCP/IP server applications register with WLM, if they are capable of doing so.

To take advantage of connection optimization even when registration is not available, consider the following scenarios.

TCP/IP Server Application Does Not Register: In some cases, you might want to use connection optimization for a particular TCP/IP server application, but it does not register with WLM. If the *stacks* that these applications are running on register with WLM, users can still use connection optimization with server applications by entering the sysplex domain name. A typical invocation might be the following:

```
tn3270 mvsplex
```

where tn3270 is the name of the invoked application and mvsplex is the sysplex domain name.

In this scenario, system administrators must ensure that equivalent instances of the server application are running on each registered stack. Otherwise, connection optimization might result in connecting the client application to a host that is not

running the server application. (mvsc in Figure 48 on page 360, for example, is not running the TN3270 server.) Depending on the client software, connection timeouts and connection retries might result.

One or More (or All) Stacks Do Not Register with WLM: In some cases, you might want to use connection optimization for a particular TCP/IP server application, but one or more (or all) stacks do not register with WLM because they do not support WLM registration or they are not configured to do so. Consider also that the server application does not register with WLM. For the stacks that have not registered, only the statically defined addresses will be used.

In this scenario, a certain degree of connection optimization can occur between hosts whose stacks register with WLM if users enter the sysplex domain name (for example, mvsplex in Figure 48 on page 360.) In addition, system administrators must ensure that equivalent instances of the server application are running on each registered stack. Connections will not be made to hosts whose stacks do not register.

Similarly, suppose that the application programmer develops a server application called ourApp that registers with the group name of myserver. Suppose also that one or more (or all) of the stacks on which the server application runs are not registered with WLM. A typical invocation might be the following:

```
ourApp myserver
```

Since one or more (or all) of the stacks are not registered with WLM, it is possible that an unusable IP address could be returned to the client because a stack has not reported the IP addresses that are active. In this case, only statically defined addresses are used for stacks that are not registered. If an unusable IP address is returned to the client, the connection times out, and depending on the client software, the client application might or might not retry the same or different address returned by the DNS query.

Considerations for Connection Balancing with Multiple Instances of TCP/IP on a Single Host System: Results of connection balancing in a multiple TCP/IP environment are not predictable. Servers such as TN3270 that bind to a single instance of TCP/IP should work as expected as long as they provide the correct host name in the WLM registration. Servers such as FTP which do not bind to a single instance of TCP/IP will have less predictable results. The host name provided on the registration will link each server with a set of IP addresses for a single stack.

Multiple Servers on the Same Port: When running multiple servers on the same port on the same TCP/IP instance (using SHAREPORT), only one of the servers should register with WLM.

Caching Issues: Proper distribution of server application addresses within a cluster requires DNS queries to be answered by the name server within the sysplex. For this reason, name servers that are located outside the sysplex cannot be configured as primary or secondary servers for the sysplex domain.

Name servers or resolvers outside the sysplex can prevent client application queries from reaching the sysplex name servers on subsequent requests if they use cached information. This is undesirable for connection optimization since the name servers and resolvers would not have up-to-date information about capacity and availability of the resources in the sysplex.

To disable other name servers from caching information about sysplex domain resources, a time-to-live (ttl) value of 0 is returned by default. Note that some resolver and name server implementations do not support a ttl value of 0 or anything less than an internally defined minimum (for example, 300 seconds).

Depending on the DNS and network configuration, the number of DNS queries on the network for the sysplex resources might increase. At the expense of reduced availability and load distribution information, administrators can choose a different default ttl for the sysplex resources by using the `-l` option when starting the sysplex name server.

Configuring a Sysplex Domain for Connection Optimization

Follow the steps below to configure name servers in a sysplex domain:

1. Identify server applications.
2. Configure server applications for WLM registration.
3. Choose sysplex name and identify name servers in the sysplex.
4. Update parent domain name server.
5. Configure the sysplex name servers.
6. Configure client applications.
7. Configure WLM in goal mode.

Each of these steps is explained below.

Step 1: Identify Server Applications: Identify the server applications to run in the sysplex. Refer to the product documentation for each application to determine if it supports registration with WLM for connection optimization. It is possible to modify the application to register with WLM. See "Registering Your Own Applications" on page 371.

Candidate applications must have the following attributes:

- Client applications must use DNS for name resolution.
- Server applications must run in a single sysplex.
- Server applications within a specified group provide equivalent functions to their clients. That is, the client application receives the same services from any of the registered server applications.
- To be considered equivalent, all servers registering in a group must be listening on the same port.
- Client applications must use portmapper or a well-known port number to access the server application.

Note: Data Facility Storage Management Subsystem (DFSMS) restrictions do not currently allow sharing of HFS files in write mode.

Maximum benefits are attained when server applications have the following attributes:

- *Registration with WLM.* This feature allows WLM to track the availability of the registering server application and to allow clusters of servers on specified systems within the sysplex. Client application use of the sysplex domain name assumes that the server application is available on all hosts within the sysplex and the stacks running on the sysplex hosts are configured to register with WLM. Even if a server application does not register with WLM, it might still be able to take advantage of connection optimization under certain circumstances. See "Usage Considerations in a Connection Optimized Sysplex" on page 364.

- *Workloads.* The server application has system or network workloads sufficient enough to warrant load distribution. Determination of what is ″sufficient″ is subjective, but the value of balancing the incoming connections should outweigh the cost of the extra DNS queries on the network. See “Caching Issues” on page 365 for more information.
- *Session Length.* In most cases, server applications selected to use the connection optimization model should have long sessions. Because reduced caching in the network name servers causes additional network traffic, server applications with many short sessions might not be appropriate candidates.

Step 2: Configure Server Applications for WLM Registration: After identifying server applications you want to run in the sysplex, you configure them for WLM registration. Typical configuration involves specification of the group name by which the application will be known. (Do not use ″TCPIP″ or the sysplex domain name as a group name.) See “Registering Your Own Applications” on page 371 for information about how to register the applications you write. The following sections describe how to configure the z/OS CS TCP/IP stack and the applications that are able to register with WLM.

Configuring TN3270: To configure TN3270 servers for registration with WLM, use the WLMCLUSTERNAME and ENDWLMCLUSTERNAME statements to enter a unique sysplex server group name (or names) in the TELNETPARMS section of the PROFILE.TCPIP data set. For a complete description of these statements, refer to *z/OS Communications Server: IP Configuration Reference*.

Unique TN3270 server group names can be used to separate or direct client application requests to only those host systems supporting the target application. For example, a TN3270 server group name of `CICS3270` could be used on host systems actually running the target CICS application that the TN3270 clients access.

If necessary, modify the server group names using the VARY OBEYFILE command. (The VARY OBEYFILE command allows changes to the system operation and network configuration without stopping and restarting the TCP/IP address space.) If using VARY OBEYFILE, however, specify *all* of the TN3270 parameters between the TELNETPARMS and ENDTELNETPARMS statements (not just additions and deletions). For more information on OBEYFILE processing, refer to *z/OS Communications Server: IP Configuration Reference*.

Configuring TCP/IP: To register TCP/IP with WLM at startup, add the IPCONFIG SYSPLEXROUTING statement in the PROFILE.TCPIP configuration data set. To register TCP/IP using VARY OBEYFILE by adding the IPCONFIG SYSPLEXROUTING statement to the obeyfile. For a complete description of this statement, refer to *z/OS Communications Server: IP Configuration Reference*.

TCP/IP also registers addresses that are active for each stack. These addresses are the active, configured addresses in the `HOME` list for which a `START` device has been processed. WLM registration supports only 15 addresses per TCPIP instance.

To deregister TCP/IP, add IPCONFIG NOSYSPLEXROUTING to the obeyfile. (Although it is possible to add this statement to the PROFILE.TCPIP data set, typical de-registration occurs in the obeyfile.) When using VARY OBEYFILE processing with IPCONFIG NOSYSPLEXROUTING, add IPCONFIG SYSPLEXROUTING on a subsequent VARY OBEYFILE command to reregister. Note that de-registration occurs automatically when TCP/IP terminates.

Note: Use the IPCONFIG SYSPLEXROUTING and IPCONFIG NOSYSPLEXROUTING statements only when the host is running as part of a sysplex.

Configuring the FTP Server: Refer to *z/OS Communications Server: IP Configuration Reference* for more information on WLMCLUSTERNAME statement.

Configuring CICS: See *OS/390 SecureWay Communications Server: IP CICS Sockets Guide*.

Step 3: Choose Sysplex Name and Identify Name Servers: Identify a unique name with which DNS client applications can access the sysplex. This name can be the same as the configured sysplex name. Names must be 18 characters or less to accommodate WLM restrictions. The sysplex name becomes part of the fully qualified domain name. For example, a sysplex, `mvsplex`, in the domain `mycorp.com` has a fully qualified name of `mvsplex.mycorp.com`. (This is the domain name you specify in the primary directive that contains the `cluster` keyword in the named boot file.)

After selecting a name for the sysplex, identify the master name servers. A sysplex should have only one primary name server, and it should have a secondary name server to provide redundancy. The secondary name server must run in the same sysplex as the primary name server. Both the primary and secondary name servers configured for connection optimization must run on hosts within the sysplex.

Step 4: Update Parent Domain Name Server: The parent domain name server is the primary name server authoritative for the domain that contains the sysplex subdomain. The parent domain name server can be the same name server as the sysplex name server.

Update the parent domain name server data files by entering the names and IP addresses of the name servers for the sysplex. Use NS (Name Server) resource records to enter the information.

For example, if the name of the domain in which the sysplex is located is `mycorp.com` and the name of the sysplex is `mvsplex` and the master name servers are running on hosts `mvsb` and `mvsc` in the sysplex, add the following records to the forward domain data file of the primary domain name server for`mycorp.com`:

```
mvsplex NS mvsb.mvsplex.mycorp.com.
        NS mvsc.mvsplex.mycorp.com.
mvsb.mvsplex.mycorp.com. A  9.67.116.201
                         A  9.67.116.206
                         A  9.67.116.208
mvsc.mvsplex.mycorp.com. A  9.67.116.203
                         A  9.67.116.207
                         A  9.67.116.210
```

This tells the parent domain name server that mvsb and mvsc are master servers for mvsplex. It does not tell the name server that mvsc is secondary, only that it is an additional master name server in the sysplex. The address (A) records are glue records. They enable remote name servers to contact the name servers in the sysplex.

In addition, it might be useful to add CNAME records for resources within the sysplex subdomain to avoid a name change in the client application and to prevent confusion for the end user. See “Step 6: Configure Client Applications” on page 370.

For example, if client applications use a default domain name of mycorp.com, then requests for `tnsysplex.mycorp.com` can be delegated to the sysplex subdomain with the following resource record:

```
tnsysplex   CNAME   tnsysplex.mvsplex.mycorp.com.
```

Step 5: Configure the Sysplex Name Servers: When the TCP/IP stack is registered with WLM, the list of IP addresses made available to client applications includes the addresses that are common to those provided by TCP/IP and the list of application servers in the forward domain data file of the sysplex name server. See "Generated Names vs. Statically Defined Names" on page 361.

Note: The addresses returned to the client application depend upon several factors including whether all or some stacks in the sysplex are registered with WLM, the availability of the adapters assigned to the IP addresses, the names the client application uses for connection, and whether the servers are registered with WLM. See "Usage Considerations in a Connection Optimized Sysplex" on page 364.

The steps for configuring sysplex name servers are similar to those for configuring the name servers in an ordinary subdomain:

"Step 2. Specify Stack Affinity (Multiple Stack Environment)" on page 335.

"Step 3. Specify Port Ownership" on page 335.

"Step 4. Update the Name Server Start Procedure (Optional)" on page 335.

"Step 5. Create the Domain Data Files (Primary Name Server Only)" on page 336 . See examples below for more information.

"Step 6. Create the Hints (Root Server) File" on page 339.

"Step 7. Create the Loopback File" on page 340.

"Step 1. Create the Boot File" on page 333. See example below for more information.

"Step 11. Start the Name Server" on page 342.

Sysplex Data Files: The data files must contain the ″A″ resource records for the host names internal to the sysplex. The host names must match the host names specified in the stack's TCPIP.DATA data set. For a discussion of how IP addresses are associated with a sysplex domain name, see "Associating IP Addresses with the Sysplex Domain Name" on page 361.

Note: Consider using VIPA addresses for the MVS hosts. If VIPA addresses are used, they are the only addresses needed to code in the forward domain data file. When Dynamic VIPAs (DVIPAs) are used for VIPA Takeover, code all of the DVIPAs in the sysplex under each host name in the forward domain data file. This will circumvent manual intervention in the DNS data files when a DVIPA is taken over or given back and will not cause any unexpected results in DNS/WLM.

Following is an example of a forward domain data file for a sysplex name server.

```
mvsplex.mycorp.com.  SOA mvsb.mvsplex.mycorp.com.
                              administrator@us.mycorp.com.  (
   1997061300  ; serial
   10800       ; refresh after 3 hours
   1800        ; retry 1/2 hour after failed zone transfer
   3600000     ; expire; length of time secondary keeps data unless refreshed
   259200 )    ; default TTL for static resource records = 3 days
;
;  Define nameservers
```

```
          IN  NS  mvsb.mvsplex.mycorp.com.
          IN  NS  mvsc.mvsplex.mycorp.com.
;
;  Define localhost
;
localhost IN  A   127.0.0.0
;
;  Hostnames specified here must match hostnames
;  specified in the stack's TCPIP.DATA file.
mvsa      IN  A     9.1.1.1
          IN  A     9.1.1.2
          IN  A     9.1.1.3
              TXT   "Text record"
              HINFO "3090" "OS/390R4"
mvsb      IN  A     9.1.1.4
          IN  A     9.1.1.5
          IN  A     9.1.1.6
          IN  A     9.1.1.7
mvsc      IN  A     9.1.1.8
          IN  A     9.1.1.9
          IN  A     9.1.1.10
```

Sysplex Boot Files: The boot file is the main configuration file for a name server. It points the name server to the domain data files, the loopback file, and the hints (root server) file. The contents and format of boot files for sysplex name servers are identical to those for the boot files of ordinary master servers with the exception of an additional keyword, `cluster`. This keyword is used only once in a boot file, at the end of either the primary or secondary directive to identify the sysplex domain.

The following is a sample boot file for the primary name server in a sysplex:

```
directory /etc/dnsdata
primary    mvsplex.mycorp.com         named.wlm.for    cluster
primary    113.67.9.in-addr.arpa      named.rev
primary    0.0.127.in-addr.arpa        named.lbk
cache      .                          named.ca
```

The file `named.wlm.for` identifies the forward domain data file for the primary name server in the sysplex.

The following is an example of a boot file for a secondary name server in a sysplex:

```
directory   /u/usr35/plex/secondary/zone1
secondary   mvsplex.tcp.raleigh.ibm.com   9.67.116.200 mvsplex.bak cluster
secondary   116.67.9.in-addr.arpa         9.67.116.200 mvsplex.rev
primary     0.0.127.in-addr.arpa           db.127.0.0
cache       .                            named.ca
```

Step 6: Configure Client Applications: Since the sysplex domain is created as a subdomain of an existing domain, resolvers should be reconfigured if it is expected that the clients will use names in the sysplex domain. Otherwise, the client will be forced to use fully qualified domain names to resolve sysplex domain names. For example, if the resolver's default domain was `mycorp.com` before adding the sysplex domain, consider changing the resolver's default domain to `mvsplex.mycorp.com`. Also consider adding the sysplex domain to the resolver's search list if the client's resolver supports it.

Note: None of the z/OS CS resolvers support search lists, with exception of nslookup's private resolver.

See “Name Resolution” on page 359 for the various names a client can specify.

Step 7: Configure WLM in Goal Mode: All hosts in a sysplex must operate in goal mode for proper load balancing (splitting). If hosts are not in goal mode, they are treated with equal weight and round-robin selection is applied.

Configure WLM in goal mode on a host by issuing the following MVS command:

```
F WLM,MODE=GOAL
```

Alternatively, IPL in goal mode by omitting the `IPS=` keyword from your IEASYSxx parmlib member and from your IEASYS00 parmlib member. See *z/OS MVS Programming: Workload Management Services*.

Registering Your Own Applications

Register a server application with WLM using a C interface or an assembler interface. See “Step 1: Identify Server Applications” on page 366. The C function is invoked as follows:

```
extern long IWMDNREG( char *group_name,
                      char *host_name,
                      char *server_name,
                      char *netid,
                      char *wlm_user_data,
                      long *diag_code);
```

A sample header file, `iwmwdnsh.h`, comes with the product. Refer to the program directory for its location. The following definitions apply:

- *group_name* is the name client applications use. It can be up to 18 characters.
- *host_name* is the TCP/IP name of the host on which the server is running. See *z/OS C/C++ Programming Guide*.
- *server_name* is a unique name that defines a particular instance of the server. It can be up to eight characters.
- *netid* and *wlm_user_data* should be null pointers.

Return values and diag_code values are documented in *z/OS MVS Programming: Workload Management Services* .

To register with a macro, use the IWMSRSRG macro. For a description of this macro, see *z/OS MVS Programming: Workload Management Services*. When using this macro, note the following:

- *Location* contains the *group_name*
- *Network_ID* can be blank.
- *LUName* contains the *server_name*.
- *Host* contains the TCP/IP host name.

You can deregister a server application from WLM using a C interface or an assembler interface. Deregister whenever you do not want the server to receive additional client application connections. The C function is invoked as follows:

```
extern long IWMDNDRG( char *group_name,
                      char *host_name,
                      char *server_name,
                      char *netid,
                      long *diag_code);
```

To deregister with a macro, use the IWMSRDRS macro. For a description of this macro, see *z/OS MVS Programming: Workload Management Services*.

For C interfaces for WLM registration and de-registration calls, see the sample registration file, SEZASAMP(WLMREG). Refer to the program directory for the location of the file.

Dynamic IP

This section describes the purpose of Dynamic IP and its benefits. Also included is an introduction to the Dynamic IP components and an overview of design concepts.

Overview

To add a new workstation to an IP network, several parameters and a variety of information is required to configure the TCP/IP software. Network components, such as a domain name server, are also required. A new TCP/IP host would normally require the following information:

- IP address
- IP subnet mask
- Default router address
- Local host name
- Domain name
- Name server address

Additional parameters, such as other server addresses, time zones, or protocol-specific configurations, might also be necessary.

Keeping track of that information in a large TCP/IP network is not an easy task for network administrators, especially if users or machines or both change their location frequently. IP address lists and domain name server databases have to be updated manually to keep track of any changes in the network.

From a user's point of view, a system administrator would have to be called to provide the necessary information to install a TCP/IP system. If the user moves to another location, this information must not be taken; the user will have to be assigned at least a new IP address if not a new host name as well. Thus, users could potentially cause disorder in a TCP/IP network.

Even if workstations will be automatically installed using software distribution techniques, the TCP/IP configuration parameters have to be pre-assigned per distribution client.

The Bootstrap Protocol (BootP), as described in RFCs 951 and 1497, was introduced to the TCP/IP community in 1985 to provide automatic assignment of some TCP/IP configuration parameters to a new TCP/IP host. A table has to be maintained at BootP servers to enter information specific to any client that has been planned for installation. Typically, clients are identified by their LAN adapter's hardware address, which has to be known to the system administrator in charge of a BootP server before he can prepare a new client entry in the database. Even though some manufacturers nowadays put the adapter hardware address on a label on the backplane of their LAN adapters, this is a tedious process if many hosts have to be installed in a short period of time.

Objectives and Customer Benefits of Dynamic IP: To lessen the problems of having to manually update any centrally maintained information files and of having a user manually configure a TCP/IP workstation, the Dynamic Host Configuration

Protocol (DHCP) has been designed and is described in RFCs 1533, 1534, 1541, and 1542. A DHCP server need not be preconfigured with a workstation's LAN address to submit the necessary TCP/IP configuration to it.

With DHCP in place, the assignment of IP addresses is a lot easier, but one problem persists—how would a domain name server learn about dynamically assigned IP addresses and host names so it can update its database accordingly? This can be solved by the Dynamic Domain Name Services (DDNS).

IBM is actively participating in the designs and implementations of DHCP and DDNS, and it has coined the term *Dynamic IP*. To summarize, the objectives of Dynamic IP and its benefits to TCP/IP system administrators and users are as follows:

- Provides automatic IP network access and host configuration
- Simplifies IP network administration
- Leverages existing IP network products and infrastructure
- Employs only open standards
- Allows customers to administer site-specific host environments
- Enables customized, location-sensitive parameter setups

Note: For further information on the Dynamic Host Configuration Protocol (DHCP) server, see "Configuring the DHCP Server for z/OS" on page 377.

Dynamic IP Components: Table 16 gives a brief description of the four types of network components that comprise Dynamic IP.

Table 16. DHCP Server Configuration

System Components	Description
Dynamic IP Hosts	Dynamic IP hosts contain DHCP client software and might contain Dynamic DNS client software. Together, they discover and cooperate with their DHCP and Dynamic DNS server counterparts in the network to automatically configure the hosts for network participation.
DHCP Servers	DHCP servers provide the addresses and configuration information to DHCP and BootP clients on the network. DHCP servers contain information about the network configuration and about host operational parameters, as specified by the network administrator. DHCP server can also be configured to be the proxy for the DDNS client and issue the commands to update the Dynamic DNS server.
DDNS Servers	Dynamic DNS servers are a superset of today's static DNS servers. The dynamic enhancements enable client hosts to dynamically register their name and address mappings in the DNS tables directly, rather than having an administrator manually perform the updates.
BootP Relay Agents (or BootP Helpers)	BootP relay agents can be used in IP router products to pass information between DHCP clients and servers. BootP relays eliminate the need for having a DHCP server on each subnet to service broadcast requests from DHCP clients.

Administering Dynamic Domains

The DDNS server performs Dynamic DNS database updates in the appropriate domain file as the updates occur. Therefore, **do not edit domain files for dynamic zones while the DDNS server is running.** Furthermore, dynamic domains cannot be dynamically reinitialized with new configuration information using the traditional **nssig -HUP** command.

Also, note that when entering comments into a domain file for a dynamic zone, the comments will be deleted when the first update to the domain is made. Domain file comments are not maintained because they would degrade the performance of the file update process.

Change the configuration information for a dynamic domain in two different ways:

- Manually, by editing the domain files **only after shutting down the DDNS server**
- Dynamically, by using **nsupdate** while the DDNS server is running

For information on how to manually enter configuration information into domain files, refer to "Step 5. Create the Domain Data Files (Primary Name Server Only)" on page 336.

When first setting up your dynamic domain, **nsupdate -g** was used to create the zone key RSA key pair; **nsupdate -g** creates an entry in the /etc/ddns.dat file. The public key, the second key in the /etc/ddns.dat file entry, needs to be copied into the appropriate domain file. The zone private key stored in /etc/ddns.dat is used by **nsupdate** when signing update requests for the administrator of the zone. The server then examines the signature to identify update requests from the zone administrator versus those from ordinary hosts. The zone key gives the possessor the power to use **nsupdate** to create, modify, and delete any host's record in a dynamic domain.

Once the zone key information is generated and the DDNS server started, the administrator can take the /etc/ddns.dat file with him and administer the zone remotely using **nsupdate**.

Migrating an Existing DNS Configuration to Dynamic IP

Before you migrate a nameserver from static DNS to dynamic DNS, you should decide if you want to:

- Leave existing resource records as they are and allow new ones to be created and updated dynamically. This will allow existing systems to keep their host names, but they will not be able to update their resource records dynamically unless a system administrator deletes them.
- Delete all existing resource records and start with a dynamic domain from the beginning.

Follow the steps below to migrate existing DNS server configuration files to Dynamic IP:

1. Modify your existing DNS configuration files to resemble the files as shown in the previous example. In the case of a boot file you have to add the *dynamic secured* or *dynamic presecured* keywords to the *primary* statements for the authoritative DNS server that is being upgraded.
2. Use the NSUPDATE -g command to create the encryption keys. Copy the public key to the zone files.
3. Start the DDNS server.
4. If you have a DHCP server configured for DDNS updates, a new entry is needed in the DDNS.DAT file for each zone the DHCP server will update.

RSA Encryption

Because the z/OS UNIX DDNS server and client products implement not only dynamic DNS but also DNS security functions, below is a brief explanation of

cryptography. This section is courtesy of RSA Data Security, Inc., Redwood City, California, and has been modified slightly here for z/OS CS.

Secret Key Cryptography: This method uses a secret key to encrypt a message. The same secret key must be used again to decrypt the message. This means that the key must be sent along with the message which exposes it to whoever might be eavesdropping on the conversation. Secret keys are very fast in terms of processing, and it is not easy to break them even though they are exposed through the communication process.

Public Key Cryptography: This method uses a combination of a modulus and a pair of exponents, called the public key and the private key. Exponents and modulus must be used together to encrypt or decrypt a message, but only the modulus and the public exponent are communicated since they are important to everyone who wants to send or receive encrypted messages using this method. The private exponent will never be publicly exposed. This ensures that no one else can decrypt messages that have been intended for a specified recipient, nor can anyone else disguise as that recipient to intercept a message.

Encryption and Authentication: Encryption means that a message will be scrambled before it can be sent over a communications link. The plain message itself will never be sent to ensure privacy. Authentication is used to ensure that a message has indeed originated from the source specified in the message, and that the message has not been altered in transit. It additionally serves the purpose of non-repudiation, which means that whoever has digitally signed a message cannot claim later that he or she has not done so. In this case, the plain message itself will be sent since there is no need for privacy. The message will also be used to generate a digital signature by using one of the aforementioned cryptographic methods, preferably public keys.

Hash Functions: A hash function is a computation that takes a variable-size input and returns a fixed-size string, which is called the hash value. If the hash function is one-way, that means hard to invert, it is also called a message-digest function, and the result is called a message digest. The idea is that a digest represents concisely the longer message or document from which it was computed; one can think of a message digest as a digital fingerprint of the larger document.

The RSA Encryption Standard: This standard public key encryption method, along with the MD5 hash function, is used with the IBM DDNS product in z/OS CS. The principle of the RSA algorithm is as follows:

1. Take two large primes, p and q.
2. Find their product n = p * q; n is called the modulus.
3. Choose a number, e, less than n and relatively prime to (p-1) * (q-1).
4. Find its inverse, d, mod (p-1) * (q-1), which means that e * d = 1 mod (p-1) * (q-1).

e and d are called the public and private exponents, respectively. The public key is the pair (n,e); the private key is d. The factors p and q must be kept secret or destroyed.

An example of RSA privacy (encryption) follows. Suppose Alice wants to send a private message, m, to Bob. Alice creates the ciphertext c by exponentiating:

```
c = m^e mod n
```

where e and n are Bob's public key. To decrypt, Bob also exponentiates:

```
m = c^d mod n
```

and recovers the original message, m; the relationship between e and d ensures that Bob correctly recovers m. Since only Bob knows d, only Bob can decrypt.

An example of RSA authentication follows. Suppose Alice wants to send a signed document, m, to Bob. Alice creates a digital signature s by exponentiating:

```
s = m^d mod n
```

where d and n belong to Alice's key pair. She sends s and m to Bob. To verify the signature, Bob exponentiates and checks that the message, m, is recovered:

```
m = s^e mod n
```

where e and n belong to Alice's public key.

Thus encryption and authentication take place without any sharing of private keys: each person uses only other people's public keys and his or her own private key. Anyone can send an encrypted message or verify a signed message, using only public keys, but only someone in possession of the correct private key can decrypt or sign a message.

To make encryption methods secure, a fairly large modulus should be chosen since it becomes increasingly difficult to break a large number into factors to determine the original primes. RSA uses a minimum length of 512 bits for the modulus, which would convert to a number with approximately 155 digits.

Due to security concerns, public key systems that use a key length of more than 512 bits must not be exported from the US.

For encryption, in reality, RSA is combined with a secret-key crypto system, such as DES, to encrypt a message by means of an RSA digital envelope. Data Encryption Standard (DES) is one of the most widely used secret key algorithms and was originally developed by IBM.

Suppose Alice wishes to send an encrypted message to Bob. She first encrypts the message with DES, using a randomly chosen DES key. Then she looks up Bob's public key and uses it to encrypt the DES key. The DES-encrypted message and the RSA-encrypted DES key together form the RSA digital envelope and are sent to Bob. Upon receiving the digital envelope, Bob decrypts the DES key with his private key, then uses the DES key to decrypt the message itself.

For authentication, in reality, RSA is combined with a hash function, such as MD5.

Suppose Alice wishes to send a signed message to Bob. She uses a hash function on the message to create a message digest, which serves as a digital fingerprint of the message. She then encrypts the message digest with her RSA private key; this is the digital signature, which she sends to Bob along with the message itself. Bob, upon receiving the message and signature, decrypts the signature with Alice's public key to recover the message digest. He then hashes the message with the same hash function Alice used and compares the result to the message digest decrypted from the signature. If they are exactly equal, the signature has been successfully verified, and he can then be confident that the message did indeed come from Alice. If, however, they are not equal, then the message either originated elsewhere or was altered after it was signed, and he rejects the message.

For authentication, the roles of the public and private keys are converse to their roles in encryption, where the public key is used to encrypt and the private key to decrypt. In practice, the public exponent is usually much smaller than the private

exponent; this means that the verification of a signature is faster than the signing. This is recommended because a message or document will only be signed by an individual once, but the signature can be verified many times.

Configuring the DHCP Server for z/OS

Dynamic Host Configuration Protocol (DHCP) allows clients to obtain IP network configuration information, including IP addresses, from a central DHCP server. The DHCP server controls whether the addresses it provides to clients are allocated permanently or are leased for a specific period. When a client is allocated a leased address, it must periodically request that the server revalidate the address and renew the lease.

The process of address allocation, leasing, and lease renewal are all handled dynamically by the DHCP client and server programs and are transparent to the end user.

DHCP defines three IP address allocation policies:

Dynamic
A DHCP server assigns a temporary, leased IP address to a DHCP client.

Static A DHCP server administrator assigns a static, predefined address reserved for a specific DHCP client.

Permanent
A DHCP server administrator assigns a permanent IP address to a DHCP client. No process of lease renewal is required.

Note: If the network uses routers or gateways, ensure that they can be enabled as DHCP relay agents. Enabling the routers or gateways for DHCP allows the DHCP packets to be sent across the network to other LAN segments.

If there are no routers that can configure to be used as DHCP relay agents, you could:

- Use a UNIX system or RS/6000® system that has the necessary code to be configured to receive limited DHCP broadcasts. Then, forward those broadcast requests to the appropriate host server.
- Use a host server that is located on the same LAN segment as the IBM Network Stations. This would eliminate any need for routers or intermediate UNIX systems to pass on the broadcast requests of the IBM Network Stations.

For dynamic address allocation, a DHCP client that does not have a permanent lease must periodically request the renewal of its lease on its current IP address in order to keep using it. The process of renewing leased IP addresses occurs dynamically as part of the DHCP and is transparent to the user.

How Does DHCP Work?: DHCP allows clients to obtain IP network configuration information, including IP addresses, from a central DHCP server. DHCP servers control whether the addresses they provide to clients are allocated permanently or are ″leased″ for a specific time period. When a client receives a leased address, it must periodically request that the server revalidate the address and renew the lease.

The DHCP client and server programs handle the processes of address allocation, leasing, and lease renewal.

To further explain how DHCP works, let's look at some frequently asked questions:

- How is configuration information acquired?
- How are leases renewed?
- What happens when a client moves out of its subnet?
- How are changes implemented in the network?

Acquiring Configuration Information: DHCP allows DHCP clients to obtain an IP address and other configuration information through a request process to a DHCP server. DHCP clients use RFC-architected messages to accept and use the options served them by the DHCP server. For example:

1. The client broadcasts a message (containing its client ID) announcing its presence and requesting an IP address (DHCPDISCOVER message) and desired options such as subnet mask, domain name server, domain name, and static route.
2. Optionally, if routers on the network are configured to forward DHCP and BOOTP messages (using BOOTP Relay), the broadcast message is forwarded to DHCP servers on the attached networks.
3. Each DHCP server that receives the client's DHCPDISCOVER message can send a DHCPOFFER message to the client offering an IP address (a server that does not want to serve a client can simply ignore the DHCPDISCOVER).

   The server checks the configuration file to see if it should assign a static or dynamic address to this client.

   In the case of a dynamic address, the server selects an address from the address pool, choosing the least recently used address. An address pool is a range of IP addresses to be leased to clients. In the case of a static address, the server uses a Client statement from the DHCP server configuration file to assign options to the client. Upon making the offer, the IBM DHCP server reserves the offered address.
4. The client receives the offer messages and selects the server it wants to use.
5. The client broadcasts a message indicating which server it selected and requesting use of the IP address offered by that server (DHCPREQUEST message).
6. If a server receives a DHCPREQUEST message indicating that the client has accepted the server's offer, the server marks the address as leased. If the server receives a DHCPREQUEST message indicating that the client has accepted an offer from a different server, the server returns the address it offered to the client to the available pool. If no message is received within a specified time, the server returns the address it offered to the client to the available pool. The selected server sends an acknowledgment which contains additional configuration information to the client (DHCPACK message).
7. The client determines whether the configuration information is valid. Upon receipt of a DHCPACK message, the DHCP client sends an Address Resolution Protocol (ARP) request to the supplied IP address to see if it is already in use. If it receives a response to the ARP request, the client declines (DHCPDECLINE message) the offer and initiates the process again. Otherwise, the client accepts the configuration information.
8. Accepting a valid lease, the client enters a BINDING state with the DHCP server, and proceeds to use the IP address and options.

To DHCP clients that request options, the DHCP server typically provides options that include subnet mask, domain name server, domain name, static route, class-identifier (which indicates a particular vendor), user class, and the name and path of the load image.

However, a DHCP client can request its own, unique set of options. For example, Windows NT® 3.5.1 DHCP clients are required to request options. The default set of client-requested DHCP options provided by IBM includes subnet mask, domain name server, domain name, and static route. For option descriptions, see "Specifying DHCP Options" on page 394.

Renewing Leases: The DHCP client keeps track of how much time is remaining on the lease. At a specified time prior to the expiration of the lease, usually when half of the lease time has passed, the client sends a renewal request, containing its current IP address and configuration information, to the leasing server. If the server responds with a lease offer, the DHCP client's lease is renewed.

If the DHCP server explicitly refuses the request, the DHCP client might continue to use the IP address until the lease time expires and then initiate the address request process, including broadcasting the address request. If the server is unreachable, the client might continue to use the assigned address until the lease expires.

Moving a Client Out of its Subnet: One benefit of DHCP is the freedom it provides a client host to move from one subnet to another without having to know ahead of time what IP configuration information it needs on the new subnet. As long as the subnets to which a host relocates have access to a DHCP server, a DHCP client will automatically configure itself correctly to access those subnets.

For a DHCP client to reconfigure itself to access a new subnet, the client host must be rebooted. When a host restarts on a new subnet, the DHCP client might try to renew its old lease with the DHCP server which originally allocated the address. The server refuses to renew the request since the address is not valid on the new subnet. Receiving no server response or instructions from the DHCP server, the client initiates the IP address request process to obtain a new IP address and access the network.

Implementing Changes in the Network: With DHCP, you can make changes at the server, reinitialize the server, and distribute the changes to all the appropriate clients. A DHCP client retains DHCP option values assigned by the DHCP server for the duration of the lease. If you implement configuration changes at the server while a client is already up and running, those changes are not processed by the DHCP client until the client attempts to renew its lease or until it is restarted.

Setting Up a DHCP Network: The following sections contain information to help you in setting up your DHCP system:

- To create a scoped DHCP network, see "Creating a Scoped Network" on page 380.
- To start the DHCP server, see "Starting the DHCP Server" on page 380.
- For tips on maintaining a DHCP server, see "Maintaining the DHCP Server" on page 381.

The IBM DHCP server provides configuration information to clients based on statements contained in the server's configuration file and based on information provided by the client. The server's configuration file defines the policy for allocating IP addresses and other configuration parameters. The file is a ″map″ that the server uses to determine what information should be provided to the requesting client.

Before starting the DHCP server, create or modify the DHCP server configuration file.

Once the DHCP server is running, you can also make dynamic changes to the configuration by modifying the configuration file and using the DHCP Server Maintenance program to reinitialize the DHCP server.

Creating a Scoped Network: You create a hierarchy of configuration parameters for a DHCP network by specifying some configuration values that are served globally to all clients, while other configuration values are served only to certain clients. Serving different configuration information to clients is often based on network location, equipment vendor, or user characteristics.

Depending on your configuration, you can specify subnets, classes, vendors, and clients to provide configuration information to different groups of clients:

- When defined globally, client, vendor or class options are available to DHCP clients regardless of their network location.

  Parameters specified for a subnet, class, or client are considered local to the subnet, class, or client. A client defined within a subnet inherits both the global options and the options defined for that subnet. If a parameter is specified in more than one level in the network hierarchy, the lowest level (which is the most specific) is used.
- Use the Subnet statement to specify configuration parameters for one subnet for a specific location in your network or enterprise.
- Use the Class statement to configure DHCP classes to provide unique configuration information from the server to clients that identify themselves as belonging to that class. For example, a group of clients can all use a shared printer or load image.
- Use a Vendor statement to provide unique configuration information to clients that identify themselves as using a specific vendor's equipment or software. Specially-defined options can be served to these clients.
- Use a Client statement in the DHCP server configuration file to serve specified options to a specific client or to exclude that client from service. You can also use a Client statement to exclude IP addresses from service.

For more information on obtaining information for a DHCP client, see "Maintaining the DHCP Server" on page 381.

Handling Errors in Configuration Files: Configuring the server incorrectly causes few, if any, warning messages. The DHCP server normally runs even when it encounters errors in the configuration file. The server might ignore the incorrect data and optionally post a message to its log.

Starting the DHCP Server:

Note: DHCPSD is installed in the /usr/lpp/tcpip/sbin directory.

To start the DHCP server, use a start procedure (found in the EZATDHDP memeber of SEZAINST), or the following form of the **dhcpsd** command:

dhcpsd [-q|-v] [-f configFile]

-q Starts the server in **quiet** mode, which means that no banner is displayed when the server starts.

-v Starts the server in **verbose** mode. Causes messages dealing with client communication to print to screen.

-f configFile
Is the name of the DHCP server configuration file. By default, the server searches for a file called dhcpsd.cfg in the directory specified by the ETC environment variable.

When starting the DHCP server with a procedure (proc), the example start proc is found in the DHCP member of the installation partitioned data set SEZAINST.

Maintaining the DHCP Server:

Note: DADMIN is installed in the /usr/lpp/tcpip/sbin directory.

To maintain a running DHCP server, IBM provides the **dadmin** command to:

- Reinitialize a DHCP server by causing the server to reread its configuration file
- Delete a lease
- Control server tracing
- Display client information
- Display IP address information
- Display server statistics

Note: Verbose mode provides additional information for debugging purposes. Verbose mode is allowed on any of the following **dadmin** command instances. Verbose is shown as a parameter in those instances where additional, more detailed information is of particular value.

Displaying dadmin Command Syntax: To display information about the command syntax, enter:

dadmin -?

Reinitializing the Running Server: If you make changes to the configuration file, you will need to reinitialize the running server to implement the changes. To reinitialize the server, use the following form of the **dadmin** command:

dadmin [[**-h**]*host*] **-i** [**-v**]

-h Specifies the host.

host
The IP address or host name of the DHCP server. If no server is specified, the local server is assumed.

-i Reinitializes the specified server.

-v Executes the command in verbose mode.

Displaying Client Information: To display information for a client ID, use the following form of the **dadmin** command:

dadmin -c*value* [**-v**]

-c Requests information for one or more clients that match this client ID.

value
The client ID is a MAC address. For example, enter 004ac77150fc. Information is returned for any matching hardware type.

-v Executes the command in verbose mode.

Displaying IP Address Information: To display information for one IP address, use the following form of the **dadmin** command:

dadmin -q*n.n.n.n* [**-v**]

-q Requests the IP address information.

n.n.n.n
The IP address of the client.

-v Executes the command in verbose mode.

Querying an Address Pool: To display information for a pool of IP addresses, use the following form of the **dadmin** command:

dadmin -p*n.n.n.n* [**-v**]

-p Requests the address pool information.

n.n.n.n
The IP address of the address pool.

-v Executes the command in verbose mode.

Controlling Server Tracing: To start and stop tracing on the DHCP server, use the following form of the **dadmin** command:

dadmin -t*value* [**-v**]

-t Specifies server tracing.

value
The value is ON to start tracing or OFF to stop tracing.

-v Executes the command in verbose mode.

Displaying Server Statistics: To display statistics information about the pool of addresses administered by the server, use the following form of the **dadmin** command:

dadmin [[**-h**]*host*] **-n***value* [**-v**]

-h Specifies the host.

host
The IP address of the DHCP server. If no host is specified, the local server is assumed.

-n Requests statistics for the server specified as *host*.

value
The value is a decimal integer indicating the number of intervals from 0 to 100. For example, a value of three returns a summary record that includes totals information, the current interval record, and the 3 most recent history records. A value of 0 returns a summary record of activity since the last summary.

-v Executes the command in verbose mode.

Statistics include:

- Discover packets processed
- Discover packets with no response
- Offers made

- Leases granted
- Negative acknowledgments (NAKs)
- Informs processed, including informs plus acknowledgments (ACKs)
- Renewals
- Releases
- BOOTP clients processed
- proxyARec updates attempted
- Unsupported packets
- Monitor requests processed

Deleting Leases: If you find that an assigned lease is not being used and you want to make the IP address available for allocation, you can delete the lease. You can only delete one lease at a time. You will be prompted to confirm deletion of the lease. To delete the lease, use the following form of the **dadmin** command:

dadmin [**-f**] [**-v**] [[**-h**]*host*] **-d** *ip_address*

-f Forces deletion of the lease without prompting.

-v Executes the command in verbose mode.

-h Specifies the host.

host
Specifies the IP address of the DHCP server. If no server is specified, the local server is assumed.

-d Deletes the lease for the specified IP address.

ip_address
The IP address for the lease to be deleted

Configuring the DHCP Server for the IBM Network Station Client: You can configure the DHCP server to be used by an IBM Network Station™. The DHCP server sets up the subnet and specifies the next bootstrap server. The IBM Network Station client can request information. The DHCP server should be configured to provide options that include subnet mask, router, domain name, and boot file name.

Changing the DHCP Configuration File

The name of the DHCP server configuration file is dhcpsd.cfg. The default location for dhcpsd.cfg is the \etc directory. In the configuration file, you create a hierarchy of configuration parameters for a DHCP network by specifying some configuration values that are served globally to all clients and other configuration values that are served only to certain clients. The information supplied to the clients is determined by the statements you use and the position of the statements in the configuration file.

Depending on your configuration, you can specify subnets, classes, vendors, and clients to provide configuration information to different groups of clients:

- When defined globally, client, vendor or class options are available to DHCP clients regardless of their network location.

  Parameters specified for a subnet, class, or client are considered local to the subnet, class, or client. A client defined within a subnet inherits both the global options and the options defined for that subnet. If a parameter is specified in more than one level in the network hierarchy, the lowest level (which is the most specific) is used.

- Use the Subnet statement to specify configuration parameters for one subnet for a specific location in your network or enterprise.
- Use the Class statement to configure DHCP classes to provide unique configuration information from the server to clients that identify themselves as belonging to that class. For example, a group of clients can all use a shared printer or load image.
- Use a Vendor statement to provide unique configuration information to clients that identify themselves as using a specific vendor's equipment or software. Specially-defined options can be served to these clients.
- Use a Client statement in the DHCP server configuration file to serve specified options to a specific client or to exclude that client from service. You can also use a Client statement to exclude IP addresses from service.
- A sample DHCP server configuration file is installed in the HFS as /usr/lpp/tcpip/samples/dhcpsd.cfg.

Editing Tips: When editing the DHCP server configuration file, keep in mind the following:

- Comments must begin with a pound sign (#).
- Class and vendor names that include spaces must be surrounded by double quotes (″).
- Statement parameters are dependent on their position. If you omit a required parameter and enter a subsequent required parameter in a statement, the server recognizes that a parameter is missing, writes an error message to a log file, and continues to read the configuration file.
- A continuation character \ indicates that the information is continued on the next line. When used within a comment, the character is treated as part of the comment and is ignored as a continuation character.
- Braces are used to specify statements that are defined within other statements.
- If a parameter is specified in more than one place, the lowest level statement (which is the most specific) is used:
  - Statements specified outside braces are considered global and are used for all addresses served by this server unless the statement is overridden at a lower-defined level.
  - Parameters specified within braces under a statement such as a Subnet statement, are considered local and apply only to clients within the subnet.
  - Definition of a parameter in a class takes precedence over definition of the parameter in a subnet.
- Class statements are not allowed inside Client statements.
- Client statements are not allowed inside Option, Vendor, or Class statements.
- Subnet statements are not allowed inside Class or Client statements.
- Vendor statements are always defined at a global level.
- Keywords are not case-sensitive. The capitalization that is used in this documentation is not required in the configuration file. However, use the convention that keywords start with a lowercase letter and subsequent ″word″ subparts start with a capital letter. For example, a keyword is proxyARec.

DHCP Server Statements:

ServerType Statement: To specify whether the server will operate only as a standard DHCP server, will perform both normal DHCP operations and PXE proxy DHCP operations, or will act only as a redirection server (PXE proxy DHCP server), use the following statement:

```
ServerType [DHCP | PXEDHCP | PXEPROXY]
```

The default value is `DHCP`, meaning the server will operate only as a standard DHCP server and will not interpret any PXE client extensions. The server will, however, still pass DHCP options and parameters to PXE clients.

PXEDHCP specifies that the server will perform both normal DHCP operations and PXE proxy DHCP operations. PXEPROXY indicates that the server will act only as a redirection server (PXE proxy DHCP server).

ImageServer Statement: To specify the siaddr field of the reply packet for a server acting as a PXE DHCP or PXE proxy only server, use the following statement:

```
ImageServer [ip address | hostname]
```

This statement separates the PXE siaddr field (the address of the BINL server) from the DHCP siaddr field. If the server is acting as a PXE DHCP server, it will use the ImageServer value to fill in the siaddr field of the reply packet. If the ImageServer statement is not specified, the siaddr field is left null for the PXE packet. A null value indicates that the BINL server resides on the same machine as the PXE proxy server. ImageServer is a global statement.

Log Statements: To enable logging by the server, use the following statements:

- Number of DHCP log files.

  `numLogFiles` *number_of_log_files*

  number_of_log_files is the maximum number of log files maintained.
- Size of DHCP log file.

  `logFileSize` *size_of_log_file*

  size_of_log_file is the size of the log file in kilobytes.
- Name of DHCP log file.

  `logFileName` *name_of_log_file*

  name_of_log_file is the name of the most recent log file.
- Type of log item.

  `logItem` *type_of_log_item*

  type_of_log_item is the type of the item to be logged. You should specify at least one log item. *type_of_log_item* can be:

  ```
  SYSERR
  OBJERR
  PROTERR
  WARNING
  EVENT
  ACTION
  INFO
  ACNTING
  TRACE
  ```

supportBootP Statement: To specify whether the server responds to requests from BOOTP clients, use the following statement:

```
supportBootP [YES | NO]
```

The default value is `NO`.

If this server previously supported BOOTP clients and has been reconfigured not to support BOOTP clients, the address binding for any BOOTP clients that was established before the reconfiguration is maintained until the BOOTP client sends another request (when it is restarting). At that time, the server does not respond, and the binding is removed.

Use this statement outside of braces.

supportUnlistedClients Statement: To specify whether the server responds to requests from DHCP clients other than those whose client IDs are specifically listed in this configuration file, use the following statement:

```
supportUnlistedClients [YES | NO]
```

The default is YES. If you specify NO, the server responds only to requests from DHCP clients that are listed (by client ID) in the configuration file.

For example:

```
client 6 10005aa4b9ab ANY
client 6 10a03ca5a7fb ANY
```

If the `supportUnlistedClients` statement is not specified or if you specify YES, the server responds to requests from any DHCP client. Use this option to limit access to addresses that are issued by this DHCP server. Listing the client IDs for all acceptable clients might take a long time.

Use this statement outside of braces.

bootStrapServer Statement: To specify whether the DHCP server specifies a bootstrap server for BOOTP clients, use the following statement:

`bootStrapServer` *address_of_bootstrap*

address_of_bootstrap is the IP address of the bootstrap server for the client.

This statement can appear at the global level, or within a Subnet, Class, or Client statement.

Option 67, Boot File Name: For clients who need a boot or must load images to initialize, use the DHCP Option 67 (Boot File Name) for the name of the boot file. The client downloads the image from the BOOTP server. For additional information about Option 67, see “Specifying DHCP Options” on page 394.

Lease Statements:

- To specify the default lease duration for the leases that are issued by the server, use the following statement:

  `leaseTimeDefault` *amount_of_default_lease_time*

  amount_of_default_lease_time is a decimal integer, followed by a space and a unit of time. The unit of time can be years, months, weeks, days, hours, minutes, or seconds.

  Default interval:
  24 hours (1440 minutes)

  Default unit:
  minute

Minimum:
180 seconds

Maximum:
-1, which is infinity

You can use this statement at the global level. To override this statement for a set of clients, use Option 51 (IP Address Lease Time) for a specific client, a class of clients, a subnet, or at the global level.

- To specify the interval at which the lease condition of all addresses in the address pool is examined, use:

  `leaseExpireInterval` *`interval_of_lease_time`*

  interval_of_lease_time is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. If you do not specify a unit of time, minutes are assumed. The value that is specified should be less than the value for `leaseTimeDefault` to ensure that expired leases are returned to the pool in a timely manner.

  Default interval:
  1 minute

  Default unit:
  minute

  Minimum:
  15 seconds

  Maximum:
  12 hours

- To specify the maximum amount of time the server holds an offered address in reserve while waiting for a response from the client, use:

  `reservedTime` *`amount_of_time_reserved`*

  amount_of_time_reserved is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. If you do not specify a unit of time, minutes are assumed.

  Default interval:
  5 minutes

  Default unit:
  minute

  Minimum:
  30 seconds

  Maximum:
  -1, which is infinity

Subnet Statement: Use the Subnet statement to specify configuration parameters for an address pool that is administered by a server. An address pool is a range of IP addresses to be leased to clients. If you configure subnets, you can set the lease time and other options for clients that use the address pool.

- Define a subnet.

  To define a subnet, use the following statement:

  `subnet` *`subnet_address`* [*`subnet_mask`*] *`subnet_range`* [(`alias=`*`subnet_name`*]

Note: The DHCP Server Configuration program uses the parameters to the right of a left parenthesis. The DHCP server parses statements to the right of a left parenthesis as comments.

subnet_address
The address of this subnet, specified in dotted-decimal notation (for example, 192.67.48.0).

subnet_mask
The mask for the subnet, specified in dotted decimal notation or in integer format. A subnet mask divides the subnet address into a subnet portion and a host portion. If no value is entered for the subnet mask, the default is the class mask appropriate for an A, B, or C class network.

- Class A network - 255.0.0.0
- Class B network - 255.255.0.0
- Class C network - 255.255.255.0

A subnet mask can be expressed either in dotted-decimal notation or as an integer between 8 and 31. For example, you can enter a subnet mask as a dotted decimal notation of 255.255.240.0 or an integer format of 20. In subnet 192.67.48.0, a mask of 255.255.240.0 implies an address range from 192.67.48.001 to 192.67.63.254. The value 20 is the total number of 1's in a mask that is expressed in binary as 11111111.11111111.11110000.00000000.

Although not required, in most configurations the DHCP server should send Option 1 (Subnet Mask) to DHCP clients. Client operation can be unpredictable if the client does not receive subnet masks from the DHCP server and assumes a subnet mask that is not appropriate for the subnet.

subnet_range
All addresses, specified in dotted-decimal notation, to be administered to this subnet. The ranges should not overlap, for example, 192.67.48.1-192.67.48.128.

Notes:

1. In the range of addresses, do not include the address of the subnet and the address that is used for broadcast messages. For example, if the subnet address is 192.67.96.0 and the subnet mask is 255.255.240.0, do not include 192.67.96.0 and 192.67.111.255 in the range of addresses.
2. To exclude an IP address in a range of address, use the Client statement (see step 391).

(alias=*subnet_name*
A symbolic name for ease in identifying a subnet.

- Define a subnet group.

  To define a subnet group, use the following statement:

  `subnet subnet_address [subnet_mask] subnet_range [label:value[/priority]]`

  label:*value[/priority]*
  Identifies subnets that are grouped together on the same wire. *value[/priority]* is a string of 1 to 64 alphanumeric characters that identifies the subnet, followed by the priority in which this subnet's address pool is used. Do not uses spaces when specifying the label. More than one subnet can have the same identifier. *priority* is a positive integer, where 1 is a higher priority than 2. If you do not specify a priority, the highest priority is assigned. If two subnets have identical priority, the subnets within a label are processed based on the physical position in the configuration file.

For example, the following two subnets are on the same wire:

```
inOrder
subnet 192.67.49.0 255.255.240.0 192.67.49.1-192.67.49.100 label:WIRE1/2
subnet 192.67.48.0 255.255.240.0 192.67.48.1-192.67.48.50 label:WIRE1/1
```

- Serve IP addresses from multiple subnets.

  To serve IP addresses from multiple subnets, use the `inOrder` or `balance` statement. The `inOrder` or `balance` statement is defined at a global level.

  inOrder *subnet_labelslist*
  subnet_labelslist is a list of labels in which each label identifies a subnet group. Each listed group is processed in order within that group. The subnet address pool with the highest priority within that group is completely exhausted before the subnet address pool with the next highest priority is used.

  balance: *subnet_labelslist*
  subnet_labelslist is a list of labels in which each label identifies a subnet group. The server provides the first IP address from the subnet that is first in the priority list, and subsequent IP addresses from each lesser-priority subnet, repeating the cycle until addresses are exhausted equally from all subnets.

  The following is an example using the `inOrder` statement. Requests for subnet group WIRE1 exhaust addresses in subnet 192.67.48.0 (WIRE1/1) first, followed by subnet 192.67.49.0 (WIRE1/2). WIRE1 and WIRE3 are not related. Requests for subnet group WIRE3 exhaust addresses in subnet 192.67.50.0 (WIRE3/1) first, followed by subnet 192.67.51.0 (WIRE3/2), and then 192.67.50.0 (WIRE3/3), which has the same subnet address as WIRE3/1 but specifies a higher address range:

  ```
  inOrder: WIRE3 WIRE1
  subnet 192.67.49.0 255.255.240.0 192.67.49.1-192.67.49.100 label:WIRE1/2
  subnet 192.67.48.0 255.255.240.0 192.67.48.1-192.67.48.50 label:WIRE1/1
  subnet 192.67.51.0 255.255.240.0 192.67.51.1-192.67.51.50 label:WIRE3/2
  subnet 192.67.50.0 255.255.240.0 192.67.50.1-192.67.50.50 label:WIRE3/1
  subnet 192.67.50.0 255.255.240.0 192.67.50.51-192.67.50.100 label:WIRE3/3
  ```

  The following `balance` statement uses all IP addresses equally in WIRE1/3 and WIRE1/4:

  ```
  balance: WIRE1
  subnet 192.67.49.0 255.255.240.0 192.67.49.101-192.67.49.200 label:WIRE1/3
  subnet 192.67.48.0 255.255.240.0 192.67.48.201-192.67.48.300 label:WIRE1/4
  ```

  A sequence of `inOrder` or `balance` statements is cumulative. For example, the statements:

  ```
  inOrder: WIRE1
  inOrder: WIRE3
  ```

  have the cumulative effect of the single statement:

  ```
  inOrder: WIRE1 WIRE3
  ```

  Note: To disable multiple subnets, comment out either the `balance` or `inOrder` processing statement or the priority.

Class Statement: Use the Class statement to specify configuration parameters for a user-defined group of clients that are administered by a server. You can use the Class statement at the global or subnet level. When the Class statement is

specified within a Subnet statement, the server only serves clients in the class that are located in the specified subnet and request the class.

For example, to create a class that is called ″accounting″ that allows member hosts to use the LPR server (Option 9) at 192.67.123.2, do the following:

- At the DHCP server, define a class that is called ″accounting″ and set the LPR server for that class to 192.67.123.2.
- At the client, configure the client to identify itself as belonging to the class ″accounting″.

When the client requests configuration information, the server sees that it belongs to the accounting class and provides configuration information that instructs the client to use the LPR server at 192.67.123.2. DHCP clients use Option 77 to indicate their class to DHCP servers.

To define a class, use the following statement:

`class` *`class_name`* [*`class_range`*]

class_name
The user-defined label that identifies the class. The class name is an ASCII string of up to 255 characters (for example, accounting). If the class name contains spaces, it must be surrounded by double quotes.

class_range
A range of addresses. To specify a range of addresses, enter addresses in dotted-decimal notation, beginning with the lower end of the range, followed by a hyphen, then the upper end of the range, with no spaces between. For example, enter 192.17.32.1-192.17.32.128.

At a global level, a class cannot have a range. A range is allowed only when a class is defined within a subnet. The range can be a subset of the subnet range.

A client that requests an IP address from a class that has used all the addresses from its range is offered an IP address from the subnet range, if available. The client is offered the options associated with the class that has used all the addresses from its range.

To assign configuration parameters such as a lease time for all clients in a class, follow the Class statement with Option statements that are surrounded by braces. For more information about options, see“Specifying DHCP Options” on page 394.

Client Statement: Use the Client statement to specify a unique set of options for a client. You can assign either a static address and configuration parameters, or configuration parameters. You can also use the Client statement to exclude an IP address from a range of available IP addresses. You can use the Client statement at the global, subnet, or class level.

- Define a client.

  `client` *`hw_type clientID client_ipaddr`* (`alias=`*`client_name`*

  hw_type
  A number that represents the hardware type of the client computer, required to decode the MAC address. For more information about hardware types, see “Hardware Types” on page 395.

clientID
Either the hexadecimal MAC address, a string such as a domain name, or a name that is assigned to the client such as the host name. If you specify a string, you are required to enclose it in double quotes and specify 0 for the hardware type.

client_ipaddr
The DHCP client's IP address in dotted-decimal notation. *client_ipaddr* must contain an address if unlisted clients are not supported.

(alias=*client_name*
A symbolic name for ease in identifying the client. Enter **alias=***client_name* immediately after a left parenthesis. This symbolic name appears in the display of the server configuration. If no name is entered, the MAC address is used.

For example, for a client (10005aa4b9ab), to reserve the static address 192.22.3.149 and also specify a lease time (Option 51) or 12 hours (43200 seconds) and a subnet mask (Option 1), use the following statement:

```
client 6 10005aa4b9ab 192.22.3.149
{
   option 51 43200
   option 1 255.255.255.0
}
```

- Specify options and assign any IP address to a client.

  To specify options for any IP address from the subnet, use the following `client` statement:

  `client` *hw_type clientID* `ANY`

  ANY
  specifies that any IP address can be assigned to the specific client ID.
- Exclude a client ID.

  To have the DHCP server exclude requests from a particular client ID, use the following `client` statement:

  `client` *hw_type clientID* `NONE`

  NONE
  specifies no IP address and no options are served to the specified client ID.

  For example:

  ```
  client 6 10005aa4b9ab NONE
  ```
- Exclude an IP address.

  To exclude one or more IP addresses from the pool of addresses available for lease, use the following statements:

  ```
  client 0 0 192.67.3.123
  client 0 0 192.67.3.222
  ```

  In this case, the hardware type and the client ID are 0. IP addresses 192.67.3.123 and 192.67.3.222 are excluded. You must specify a separate statement for each address you want excluded.
- Exclude a range of IP addresses.

  To exclude a range of IP addresses from the pool of addresses available for lease, specify many `client` statements.

  Each range of excluded addresses should contain no more than 10 addresses. Each excluded address results in a separate `client` statement in the

configuration file. To exclude larger numbers of addresses, define subnets that do not include the addresses you want excluded. For example, to exclude addresses 50-75 in subnet 192.67.3.0, specify:

```
inOrder: WIRE1
subnet 192.67.3.0 255.255.240.0 192.67.3.1-192.67.3.49 label:WIRE1/1
subnet 192.67.3.0 255.255.240.0 192.67.3.76-192.67.3.100 label:WIRE1/2
```

Vendor Statement: To provide vendor configuration information to the DHCP clients in your network:

- Define a vendor and assign the appropriate configuration values. Unlike the Class statement, you cannot control the scope of the Vendor statement by its placement in the file. Use the Vendor statement only at the global level. Vendor statements within Subnet, Class, or Client statements are ignored. Options can be redefined in the vendor class.
- The DHCP client identifies itself to the DHCP server as belonging to a vendor class by sending Option 60 (Class Identifier) with a specific vendor name.
- The DHCP server recognizes that the client has a specific vendor and returns encapsulated Option 43 (Vendor-specific Information), which contains vendor-specific DHCP options and option values.

To define a vendor, use the following statement:

```
vendor vendor_name [hex value]
```

vendor_name
The user-defined label that identifies the vendor. The vendor name is an ASCII string of up to 255 characters (for example, ″IBM″). If the vendor name contains spaces, it must be surrounded by quotes (″).

[hex *value***]**
value must be specified either as an ASCII string or as hexadecimal in the hexadecimal ASCII string construct. For example:

```
hex "01 02 03"
```

For more information, see descriptions of Option 60 (Class-Identifier) in “Specifying DHCP Options” on page 394.

The vendor statement can also be specified in the DHCP server configuration file as a vendor statement followed by a pair of braces containing the options particular to this vendor. Within these braces, the usual option value encoding and decoding rules do not apply.

Statements Specifying Server Information:

- Querying in-use address.

  Before the server allocates an IP address, it PINGs the address to make sure that it is not already in use by a host on the network. The server places an in-use address in a special pool and allocates a different address.

  To specify the amount of time a DHCP server holds an in-use address in a special pool before returning the address to the active pool available for assignment, use the following statement:

  ```
  usedIPAddressExpireInterval in_use_time_value
  ```

  in_use_time_value is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. The default is 1000 seconds. If you do not specify a unit of time, minutes are assumed.

Default interval:
1000 seconds

Default unit:
minute

Minimum:
30 seconds

Maximum:
-1, which is infinity

- Transform canonical addresses.

  For 802.3 clients, use the canonical keyword to instruct the DHCP server to transform MAC addresses to canonical (byte starts with least significant bit) form. In most cases, you do not want the DHCP server to transform canonical addresses. MAC addresses of 802.3 clients are normally in canonical format on an 802.3 network. When 802.3 MAC addresses are transmitted across a transparent bridge, the bridge reformats the bits that identify an 802.3 client MAC address to a noncanonical (byte starts with most significant bit) form. When the bridge returns the MAC address to an 802.3 network, the bridge again reformats MAC addresses.

  To cause the DHCP server to transform MAC addresses, use the following statement:

  ```
  canonical [YES | NO]
  ```

  NO prevents the DHCP server from transforming MAC addresses. YES causes the DHCP server to transform MAC addresses. NO is the default value.

- Specify statistics snapshots.

  To specify the number of intervals that expire before the DHCP server takes a snapshot of statistics, use the following statement:

  ```
  statisticSnapshot number_of_intervals
  ```

  The length of each interval is determined by the `leaseExpireInterval` keyword. For example, a value of 3 collects statistics after a span of three intervals, where each interval has a length specified by the `leaseExpireInterval` keyword. If no value is specified, the server takes a snapshot of statistics at the end of every lease expire interval.

Additional Options: To assign additional configuration parameters, use the Option statement. All clients inherit all globally-defined options. A client that is defined within a Subnet statement inherits global options and options that are defined for that address pool. To assign configuration parameters for all clients in a subnet, follow the Subnet statement with Option statements that are surrounded by braces.

A Sample DHCP Server Configuration Files: The following is a sample DHCP configuration file:

```
logFileName dhcpsd.log
logFileSize 100
numLogFiles 4
logItem SYSERR
logItem ACNTING
logItem OBJERR
logItem EVENT
logItem PROTERR
logItem WARNING
logItem INFO
logItem TRACE
logItem ACTION
supportBootP yes
```

```
supportUnlistedClients NO
bootStrapServer 192.168.1.4
option 211 "nfs"
option 212 "10.1.1.2
option 213 "/usr/lpp/nstation/standard/configs/
option 214 "nfs"
option 67 /usr/lpp/nstation/standard/kernel

leaseTimeDefault 12 HOURS

option 15  mycompany.com

# Addresses 8.67.112.24 through 8.67.112.25 do not inherit
# options defined for 8.67.112.26 through 8.67.112.30

subnet 192.168.1.00 255.255.255.0 192.168.1.1-192.168.1.100
 {
  option 1 255.255.255.0
  option 3 10.1.1.1

  client 0 0 192.168.1.4
  client 0 0 192.168.1.5
 }
```

Specifying DHCP Options: DHCP allows you to specify options, also known as BOOTP vendor extensions, to provide additional configuration information to the client. RFC 2132 defines the options that you can use. Each option is identified by a numeric code.

Architected Options 0 though 127 and Option 255 are reserved for definition by the RFC. The DHCP server and the DHCP client use options in this set. The administrator can modify some architected options. Other options are for exclusive use by the client and server. The administrator cannot or should not configure the following options at the DHCP server:

- 52 (Option Overload)
- 53 (DHCP Message Type)
- 54 (Server Identifier)
- 55 (Parameter Request List)
- 56 (Message)
- 57 (Maximum DHCP Message Size)
- 60 (Class Identifier)

Options 128 through 254 represent options that can be defined by administrators to pass information to the DHCP client to implement site-specific configuration parameters. Additionally, IBM provides a set of IBM-specific options, such as Option 192 (TXT RR).

The format of user-defined options is:

`option` *code value*

code can be any option code 1 through 254. *value* must always be a string. At the server, it can be an ASCII string or a hexadecimal string. At the client, however, it always appears as a hexadecimal string that is passed to the processing program.

The server passes the specified value to the client. You must, however, create a program or command file to process the value. For more information about data formats for the DHCP options, refer to *z/OS Communications Server: IP Configuration Reference*.

Option Categories: The DHCP options are divided into the following categories:

- Base Options
- IP Layer Parameters per Host Options
- IP Layer Parameters per Interface Options
- Link Layer Parameters per Interface Options
- TCP Parameter Options
- Application and Service Parameter Options
- DHCP Extensions Options
- Load Balancing Options
- IBM-Specific Options

Hardware Types: The following is a list of hardware types you can specify on the Client statement:

Type	Description
0	Unspecified. If you specify a symbolic name for the client ID, specify 0 for the hardware type.
1	Ethernet (100MB)
6	IEEE 802 Networks (which include 802.5 token ring)

Configuring DHCP Server as DDNS Client Proxy

The DDNS proxy A-record update feature supports clients that do one of the following:

- Use DHCP option 81 to communicate their host name information to a DHCP server.
- Send some or all of their host name information in DHCP option 12 and option 15.
- Have their host name information defined to the DHCP server by an administrator.

The proxyARec update feature allows non-Dynamic IP clients (clients which have DHCP client capability but not DDNS capability) to effectively participate in the Dynamic IP system. This alternative is well-suited for network environments where client hosts are not mobile or do not change administrative domains.

New keywords in the DHCP server configuration file instruct the DHCP server how to interact with the DDNS server using A records on behalf of DHCP clients. The new keywords are:

proxyARec
: For more information on enabling DHCP server A record updates, see “Enabling Dynamic A Record Updates” on page 396.

updateDNSA
: For more information on enabling A record updates, see “Specifying the Keyword for A Record Updates” on page 397.

updateDNSP

For more information on enabling PTR record updates, see “Specifying the Keyword for PTR Record Updates” on page 398.

Enabling Dynamic A Record Updates: Once the DHCP server reads the configuration file and is instructed how and when to perform a DDNS proxy A record update, the following rules of precedence are used to derive the host name data to be used:

1. As a first priority, option 81 is included in the DHCP client request and indicates that an A record update should be performed at the DDNS server on behalf of the client.
2. Either option 12 or option 15, or both, are included in a DHCP client request.
3. Either option 12 or option 15, or both, are defined in the DHCP server configuration file.

The data used in the DDNS update packet is derived using one or more of the above means until a fully-qualified host name data is established. For example, in the case of Microsoft® Windows 3.11+ and Windows 95 clients, only option 12, the Microsoft Windows client computer name, is included in DHCP lease requests. In this case, the administrator can define an option 15, domain name, to use with the host name provided by a client’s option 12 to obtain the host’s fully-qualified name.

The proxy A-record update feature works with any client which is capable of including host name info in option 81 and/or options 12 and 15. Otherwise, the DHCP client host, as identified by its LAN adapter’s MAC address, and its host name information can be specified entirely in the DHCP server’s configuration file.

Note: If you use DDNS proxy A record support for subnets which include some hosts capable of updating their own DDNS A-records such as IBM OS/2 Warp Dynamic IP clients, you might want to disable those client’s own DDNS update feature by commenting out updateDNSA and UpdateDNSTxt keywords in the client’s DHCPCD.CFG configuration file. If you do not disable the dynamic A record updates at the clients, the proxy A record updates attempted at the DHCP server for those clients will fail because the DHCP client, rather than the DHCP server, will own the client’s entry on the DDNS database.

Specify the proxyARec keyword to enable the DHCP server to dynamically update an A record mapping (host name-to-IP-address) on a DDNS server for clients unwilling or unable to update their A records. The format of the proxyARec keyword is:

```
proxyARec
```

The DHCP server performs updates of the client’s A record regardless of the client’s client ID value.

The proxyARec keyword is required to be enabled globally or within the subnet, class, or client level. The updateDNSA keyword must also be specified. If the proxyARec keyword is not specified, clients will not have their A records updated.

Specify the proxyARec statement within braces to indicate the information applies only to the subnet, class, or clients that meet the criteria of the preceding conditional statement. To globally assign proxyARec, specify the keyword outside any braces.

Specifying the Keyword for A Record Updates: To specify how the server updates A records for a non-IBM IP client, use keyword:

`updateDNSA` *`command_string`*

The following is a typical updateDNSA string issued by the DHCP server:

```
updateDNSA "nsupdate -f -h%s -s"d;a;*;a;a;%s;s;%s;3110400;q" -q"
```

This default string is normally adequate. Typical changes to the command string are to modify the value of an expiration time (such as 3110400 seconds) beyond the A record expiration. Assuming proxyARec is specified, this example instructs the DDNS name server to delete all A records for this host name and add a record that maps the host name to the specified IP address for the specified lease time.

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-h Client host name (value is substituted by the DHCP server).

-s Indicates that the command should be run in command-line mode and all subcommands are included within the quotation marks that follow. The following command string is appropriate for most cases. The string contains:

d;a;*; Delete all A (host name-to-IP-address) records for this host.

a;a;%s;
Add an A record using an IP address (%s) provided by the server, where %s indicates string substitution.

s;%s; Send a lease time (%s) (value provided by the server for the IP address), where %s indicates string substitution.

3110400;
The effect of this value is to reserve the host name for an additional time interval after the A record expiration. In this case, an interval of 36 days (3110400 seconds) is added to the A record expiration time. This value works to preserve a user's host name during times when the name-to-address mapping might expire, such as holidays, vacation, or other times of inactivity.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

For more information on client control of A record updates for non-dynamic DHCP clients, see client option 81 in Specifying DHCP Options.

Releasing a Client A Record: Use the releaseDNSA keyword at a global level as a template which tells the DHCP server how to release a client record, when the client requests release. The template is adequate for normal purposes, but can be modified if necessary.

The keyword is:

`releaseDNSA` *`string`*

The following is an example of a releaseDNSA string issued:

```
releaseDNSA "nsupdate -f -h%s -s"d;a;%s;s;%s;0;q" -q"
```

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-h%s
Client host name (value is substituted by the DHCP server).

-s Information in the command string releases the DHCP client's A record at the DDNS server. The command string is appropriate for most cases. The string contains:

d;a;%s;
Delete the A (host name-to-IP-address) record for this host.

s;%s; Send a lease time (%s) (value provided by the server for the IP address).

0; An additional time interval added to the normal expiration of the host name beyond the expiration of the A resource record. In this case, the value is zero because the A resource record is deleted.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

Specifying the Keyword for PTR Record Updates: DHCP servers ″own″ the PTR records and can update DDNS servers with a PTR record (resource records that map an IP address to a host name) for each address allocated to a host that identifies itself with DHCP options 12 (host name) and 15 (domain name), or with option 81, Client DHCP-DNS.

To enable DHCP server PTR record updates, use:

```
updateDNSP command_string
```

The following is a typical updateDNSP string issued by the DHCP server:

```
updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q" -q"
```

This default string is normally adequate. Typical changes to the command string are to modify an the name expiration extension time such as 0. This example instructs the DDNS name server to delete all PTR records for this address and add a record that maps the IP address to the new host name for the specified lease time. The updateDNSP command is issued when the DHCP client is assigned an address.

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-r%s
The IP address which is to be mapped to the new host name.

-s The command should be run in command-line mode and all subcommands are included within the quotation marks that follow. The command string is appropriate for most cases. The string contains:

d;ptr;*;
Delete all PTR (IP-address-to-host name) records for this IP address.

a;ptr;%s;
Add a PTR record which maps the IP address to the host name (%s) provided by the server.

s;%s; Send a lease time (%s) (value provided by the server).

0 The effect of this value is to reserve the IP address entry this many seconds beyond when the IPaddress-to-host name mapping expires. In this case, no additional time is added.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

Note: The updateDNSP keyword is equivalent to the updateDNS keyword in earlier releases. The updateDNS keyword continues to be supported.

For more information on how a non-dynamic IP client updates its A and PTR records, see *DHCP and Dynamic IP Introduction*. For more information about the nsupdate command and its other parameters, see *DNS Administration*.

Releasing a Client PTR Record: Use the releaseDNSP keyword at a global level as a template which tells the DHCP server how to release the client PTR record, when the client requests release. The template is adequate for normal purposes, but can be modified if necessary. For example, this DHCP server keyword enables immediate release of a PTR record when a mobile client's lease is terminated.

The DHCP server command is:

```
releaseDNSP string
```

The following is an example of a releaseDNSP string:

```
releaseDNSP "nsupdate -f -r%s -s"d;ptr;%s;s;%s;0;q" -q"
```

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-r%s
The IP address substituted by the DHCP server.

-s Information in this command string releases the DHCP client's PTR (IP-address-to-host-name mapping) record at the DDNS server. The command string is appropriate for most cases. The string contains:

d;ptr;%s;
Delete the PTR record for this address.

s;%s; Send a lease time (%s) (value provided by the server for the IP address).

0 An additional time interval added to the normal expiration of the PTR resource record beyond the expiration of the PTR resource record. In this case, the value is zero because the record is deleted.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

Defining DHCP Proxy Authority

Defining DDNS support also includes ensuring the DHCP domain is recognized at the DDNS server and setting up the necessary security for making updates to the DDNS server. To enable A record and PTR record dynamic update support in your DHCP server:

- Ensure that the administrator of the DDNS server configures dynamic reverse domains (*.in-addr.arpa) for all addresses for which the DHCP server will be making dynamic PTR updates.
- Create key file entries in the DDNS.DAT file for each dynamic reverse domain to be updated. If you use DDNS in your network, you must also create a DDNS.DAT file, which contains the security keys used to process IP address and host name updates with a DDNS server. For more information about the DDNS server, see DNS Administration.

Defining DDNS Key Files for the PTR Record: The DDNS.DAT file contains the security key pairs to be used for processing IP address-to-host name and host name-to-IP-address updates for the specified addresses which are sent to the corresponding primary DDNS server. The file must contain at least one entry per primary DDNS server.

Run the nsupdate command on the server to create key entries to the /etc/ddns.dat file for DHCP supported clients. The format of the nsupdate command for creating security key pairs for updating PTR records in the DDNS.DAT file is:

```
nsupdate -f -hinverse_ip_address -g -pprimary_ddns_server
```

The command string includes:

-f The request originates from a DHCP server.

-h The inverse IP address(es) for which keys are generated.

inverse_ip_address
The format of the inverse_ip_address is the IP address in reverse with in-addr.arpa appended. For example, 9.67.149.111 as an inverse_ip_address entry is 111.149.67.9.in-addr.arpa. You can use a wildcard to expand the scope of the entry. For example, if you want this entry to apply to all hosts whose IP address begins with 9.67, you could specify *.67.9.in-addr.arpa. If you want this entry to apply to all hosts, you could specify *.in-addr.arpa.

-g A key pair should be created for this entry.

-p The primary DDNS server.

primary_ddns_server
The host name or IP address of the primary DDNS server for the specified addresses.

For example:

```
nsupdate -f -g -h *.33.37.9.in-addr.arpa -p ns-updates.company.com
```

The wildcard (*) allows aggregate entries such as *.33.37.9.in-addr.arpa to represent all addresses beginning with the three octets 9.37.33.

The format of the entry in the DDNS.DAT file for a DHCP server is:

```
inverse_ip_address  primary_ddns_server  private_key  public_key
```

The inverse_ip_address and primary_ddns_server are explained above.

The private_key and public_key are used to provide fail-save authentication for updates submitted by this DHCP server. As stored in the DDNS.DAT file, the private key is encrypted.

Defining DDNS Key Files for the A Record: The format of the nsupdate command for creating security key pairs for updating A records in the DDNS.DAT file is:

```
nsupdate -f -g -hfully_qualified_host_name -pprimary_ddns_server
```

The command string includes:

-f The request originates from a DHCP server.

-h The host name(s) for which keys are generated.

fully_qualified_host_name
The format of the fully_qualified_host_name is the host name followed by the full-qualified domain name or a wildcard to allow aggregate entries.

-g A key pair should be created for this entry.

-p The primary DDNS server.

primary_ddns_server
The host name or IP address of the primary DDNS server for the specified addresses.

For example:

```
nsupdate -f -g -h *.city.company.com -p ns-updates.company.com
```

You can specify a specific host, such as myhost, or use a wildcard to allow aggregate entries such as *.city.company.com to represent all hosts in the zone.

Configuring the BINL Server

The BINL server (binlsd) is similar to the DHCP server and is started and configured similarly. Port 4011 must be reserved for the BINL server. To reserve port 4011 for binlsd, add the following line to the PORT statement of the TCPIP profile:

```
4011 UDP BINLSD
```

Also, if the system is running in a multiple stack (common INET) environment, the INADDRANYPORT range cannot include 4011.

Following are the command line options that can be used with binlsd:

Option Description

-? Display the help message.

-b Display the program banner.

-q Execute in quiet mode.

-v Execute in verbose mode.

-f Override default configuration file location.

The BINL configuration file, dhcpsd.cfg by default, is searched for first in the current working directory. If not found in the current working directory, the /etc directory is searched. Following is a sample BINL configuration file shipped as SEZASAMP(BINLSD):

```
#########################################################################
#
#   binlsd.cfg -- BINL (Image Server) Configuration File
#
#   This file contains the directives that can be specified by the
#   server's administrator to configure the server and enforce
#   policies. This file is only a sample.  The finished file must be
#   placed in the directory specified by the ETC environment variable.
#
#   Do not put any long line without spaces in this file.
#
#   A line starting with a '#' character is a comment and is ignored.
#   A '#' on a line which is not part of a quoted string indicates
#   that anything to the right of this character is a comment and should
#   be ignored.
#
#   A continuation character of '\' is supported.  It must be
#   the last non-whitespace character on the line prior to any comments.
#
#   The directives are specified in the form of
#   <keyword><value1>...<valueN>.
#
#   Here is a partial list of keywords whose value can be specified
#   in this file:
#
#   Keyword             Effect
#   -------------       ------------------------------------------------------------
#   sa                  Specifies the system architecture.
#   nit                 Specifies the network interface type.
#   lsa1nic             Specifies the lsa1 nic type.
#   tftp                Specifies the address of the tftpd server.
#   bpname              Specifies the install filename given to clients.
#   numLogFiles         The number of log files desired.
#   logFileSize         The size of log files in kilobytes.
#   logFileName         The name of the most recent log file.
#   logItem             An item to be logged.
#   servername          The LCM server name for LSA-1 clients. Not used for LSA-2 clients.
#   serverdomainname    The LCM domain name for LSA-1 clients. Not used for LSA-2 clients.
#
#   client              Definition of a set of options for a specific client
#                       or a definition of a client not to be serviced.
#
#   pxevendor           Configuration delimiter to indicate pxe options to follow.
#
#   pxeoption           A pxe configuration option value to pass to clients.
#
#   The scope of a keyword is limited by a pair of curly brackets ({, })
#   within which the keyword is located.  If a keyword is located outside
#   of any pair of curly brackets, its scope is applicable to all the
#   clients served by this server.  The curly brackets must appear alone on
#   a line.
#
#  Log files.  This set of parameters specifies the log files that will be
#  maintained by this server.  Each parameter is identified by a keyword
#  and followed by its value.
#
#  Keyword      Value           Definition
#  --------     ------------    ------------------------------------------
#  numLogFiles  0 to 99         number of log files.  If 0 is specified,
#                               no log file will be maintained and no log
#                               message is displayed anywhere.  When a log
#                               file reaches maximum size, a new log file
#                               is created, until the maximum number of
#                               log files have been created.  Only the most
#                               recent n log files are kept/
#
#  logFileSize  in K bytes      maximum size of a log file.  When the size
```

```
#                               of the most recent log file reaches this
#                               value, it is renamed and a new log file is
#                               created.
#
#  logFileName  file path       name of the most recent log file.  Less
#                               recent log files have the number 1 to
#                               n-1 appended to their names; the larger
#                               the number, the less recent the file.
#
#  logItem                      An item that will be logged.
#               SYSERR          System error, at the interface to the platform.
#               OBJERR          Object error, in between objects in the process.
#               PROTERR         Protocol error, between client and server.
#               WARNING         Warning, worth of attention from the user.
#               EVENT           Event occurred to the process.
#               ACTION          Action taken by the process.
#               INFO            Information that might be useful.
#               ACNTING         Accounting information on clients served.
#               TRACE           Code flow, for debugging.
#
#

#  servername   <servername>
#                               If present this name is sent to LSA-1 clients in the offer
#                               packet.  If this option in not present the server will send
#                               its name to the client as the LCM server.
#  serverdomainname <domainname> <workgroup|domain>
#                               Specifies the domain name of the LCM server that is sent to
#                               LSA-1 clients in the offer packet.
#
#                               <domainname> the name of the domain
#
#                               The last parameter can be the string "workgroup" to indicate
#                               that <domainname> is a workgroup or the string "domain" to
#                               indicate that <domainname> is a domain.

#  client <id_type><id_value><exclude|value>
#                               Definition of a client record.
#
#                               <id_type> is one of the hardware types defined
#                               in RFC 1340 (e.g. 1 for 10 megabit Ethernet,
#                               6 for 802.5 Token Ring.)  The type may be
#                               0, in which case the hardware type is not
#                               specified and the
#                               id_value may be a string of any format.
#
#                               <id_value> is a character string if the type
#                               is 0. Typically, this would be a domain name.
#                               For a non-zero <id_type>, the <id_value> is
#                               a hexadecimal string representing
#                               the hardware address of the client.
#
#                               The last parameter can be blank to indicate that
#                               the client matching <id_type> and <id_value>
#                               should get the specific parameters defined in
#                               its scope.
#
#                               The last parameter can be the string "exclude" to
#                               indicate that the client matching
#                               <id_type> and <id_value> should
#                               not be serviced by this server.
#
#       The client statement may be immediately followed
#       by a pair of curly brackets, in which the options
#       particular to this client can be specified.
#
#       Note: All clients inherit all globally defined options.
```

```
#
#
#
#  Pxeoption.  This keyword identifies an option statement.  The options assigned
#  are defined in the "Wired for Management Baseline Version 1.1" specification
#  authored by the Intel Corporation.
#
#  An option is specified by the "pxeoption" keyword followed by the option code
#  of this option and its data field, in a single line.  One or more options
#  may be specified.
#
#  The scope within which an option applies is delimited by a pair of curly
#  brackets ({, }) surrounding the option statement.
#
#  If two or more options with the same option code are specified, the
#  one with the most specific scope is used.  This allows, for example,
#  an option specified at the sa and nit scope to override that same option
#  specified at the global scope.  If two or more options
#  with the same option code are specified within the same scope,
#  the first one read by the server will be the one used, (subject to
#  its being overridden by the same option in a more specific scope).
#
#  All options specified will be sent to the client encapsulated in DHCP option 43.
#

#  Sa.  This parameter specifies the system architecture.  At the global level sa is
#  considered to be "unspecified" and any sa which is received will match if a more
#  specific sa is NOT explicitly configured.  A specific sa may be configured and values
#  for boot path name, TFTP server, and options maybe scoped under this specific
#  specification.
#
#  NIT. This parameter specifies the network interface type. Works in a similar manner as
#  the system architecture (Sa.)  It can be configured to more specifically qualify a SA
#  (e.g sa 0 nit 1) or if configured alone will be considered to more specifically qualify
#  the unspecified SA.
#
#  LSA1NIC.  This parameter specifies the network interface type for LSA-1 clients.  Works
#  in a similar manner as the system architecture (Sa.) and network interface type (NIT.)
#  Specific lsa-1 nic types to provide values for boot path name and TFTP server.  No
#  additional options are sent to the LSA-1 client.
#
#
#  TFTP.   This parameter specifies the address of the tftp server that contains
#  the install image.
#
#  Keyword             Value                 Definition
#  -----------   ----------------------    -------------------------
#  tftp          [ipaddress|hostname]       If "ipaddress" actual address of image server.
#
#                                             If "hostname" dns lookup is done to resolve
#                                               ipaddress of image server.
#
#
#  bpname.  The fully qualified pathname of the install image file.
#
#  Keyword           Value
#  -----------     -------------------
#  bpname         [filename]
#
#
################################################################################

# The remaining portion of this file is an sample configuration file.
# Comments are added to assist in understanding the configuration file.
# Further information and detail is found in the online user documentation.

# Setup of the log file information.  This includes the size and name of the
```

```
# logfile along with number of logfiles maintained and type of information that
# will be logged.
numLogFiles     10
logFileSize     1000
logFileName     binlsd.log
logItem         SYSERR
logItem         OBJERR
logItem         PROTERR
logItem         WARNING
logItem         EVENT
logItem         ACTION
logItem         INFO
logItem         ACNTING
logItem         TRACE

# default TFTP server
tftp   9.33.44.55

# Fully qualified boot Image pathname
bpname c:\tftp\lccm\lccm.1

# Global Options
pxevendor
{
  pxeoption 2 555
  pxeoption 3 544
  pxeoption 4 5
  pxeoption 5 5
}
#Excluded client list
client 1 080aab4567 exclude
client 1 000abc45d7 exclude
client 6 080aab4899 exclude

sa 0 nit 1
{
  tftp 9.180.71.79
  bpname c:\tftp\lccm\lccm.1

  #Options specific to this sa and nit values
  pxevendor
  {
    pxeoption 1 224.1.1.1
    pxeoption 2 1758
    pxeoption 3 1759
    pxeoption 4 1
    pxeoption 5 2
  }
}

nit 2
{
  tftp 9.180.71.79
  bpname c:\tftp\lccm\lccm.1

  #Client excluded from service if it falls into this category.
  client 6 08aa343bf3 exclude

  # Special configuration for client following into this nit2 type
  client 1 12345abcd34
  {
    tftp 9.37.56.2
    bpname D:\intel\nit2\nt40s
  }
}

lsa1nic 53
```

```
{
  tftp 9.180.71.79
  bpname  d:\lcm\system\images\53.X

  # Special configuration for client following into this nic type
  client 1 12345abcd34
  {
    tftp 9.37.56.2
    bpname D:\intel\nic2\nt40s
  }

  # Deny this client LSA1 service
  client 6 08005aab6abc4 exclude

}

#
# end of binlsd.cfg
#
```

Configuring a DDNS Server

There is no explicit configuration utility for the DDNS server as there is for the DHCP server. You can either create new DDNS server configuration files, or you can migrate an existing DNS configuration to dynamic DNS server configuration files.

There are three ways to use the DDNS server:

- Static DDNS server
- Dynamic secured DDNS server
- Dynamic presecured DDNS server

When used as a static DDNS server, there is nothing you have to do but use your existing DNS configuration files with the DDNS server. It will then work exactly the same way as the previous DNS server.

When used in dynamic secured mode, the DDNS server will allow clients to update their resource records dynamically using encryption keys that have been created by the clients themselves.

When used in dynamic presecured mode, the DDNS server will only allow those clients to update their records to which an encryption key has been provided that has been generated by the administrator.

Creating a New DDNS Server Configuration: The files required for a minimum configuration are:

- The nameserver boot file contains the path and file names for any other configuration files. The default for this file is /etc/named.boot. It will be examined by the DDNS server at startup.
- The nameserver domain file contains information about the zones for which this server will be authoritative, and all mappings from names to IP addresses (ordinary or forward name resolution). The file name is defined in the nameserver boot file.
- The nameserver reverse file contains information about the mappings from IP addresses to names (inverse or reverse name resolution). The file name is defined in the nameserver boot file.

Use the following steps to create DDNS server configuration files from scratch:

1. Create the DDNS configuration files. You can also modify the samples that are shipped in the /usr/lpp/tcpip/samples directory.

   A nameserver boot file might look as follows:

```
;
; boot file for name server configuration.
;
directory /test/dns/zones/
;
; type       domain                     source file or host
;
primary  test.itsc.raleigh.ibm.com        test.data       dynamic secured
;
primary  200.200.200.in-addr.arpa        db.200.200.200  dynamic secured
;
```

   On the primary statements, you can specify if you want to use the DDNS server in dynamic secured or in dynamic presecured mode by using either the *dynamic secured* or the *dynamic presecured* keywords. A nameserver domain file might look as follows:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
@   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
             ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
    IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
localhost    IN A     127.0.0.1
;
ns-updates   IN A     200.200.200.2
martin       IN CNAME ns-updates
;
BPClient     IN A     200.200.200.14
;
```

   A nameserver reverse file might look as follows:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
200.200.200.in-addr.arpa   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
                                    ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
200.200.200.in-addr.arpa.  IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
;
; Addresses for the canonical names
;
2          IN PTR martin.test.itsc.raleigh.ibm.com.
14         IN PTR BPClient.test.itsc.raleigh.ibm.com.
;
```

2. After you have created the files, use the NSUPDATE -g command to create the encryption key pairs for the zone resource records in the domain and reverse files.

   After the administrator has copied the public key into the files, they might look as follows:

   - Domain Name file:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
@   IN  KEY     80 0 1 AQPOzUYWvAUyZhYxogDcrtxOZOH33V31Tmrs1Db1WYiyI4Y
                       7Mmoz6Vm3XY/QTMHOyeHcVAMKmuba+rW4/+IkMeP3
@   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
             ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
    IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
localhost    IN A     127.0.0.1
;
ns-updates   IN A     200.200.200.2
martin       IN CNAME ns-updates
;
BPClient     IN A     200.200.200.14
;
```

   - Reverse Name file:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
200.200.200.in-addr.arpa   IN  KEY     80 0 1 AQPR+3ObXCgcjmBfKSnN4fD6v
                                              VH/AUIwincGNeD1MAuz2BTQSQ
                                              /bJkXLA3nxfV+HxKfxWptkRck
                                              wzxEk1DD3DSB
200.200.200.in-addr.arpa   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
                                    ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
200.200.200.in-addr.arpa.  IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
;
; Addresses for the canonical names
;
2          IN PTR martin.test.itsc.raleigh.ibm.com.
14         IN PTR BPClient.test.itsc.raleigh.ibm.com.
;
```

   The NSUPDATE -g command will also create the DDNS.DAT file that contains the private encryption keys to sign any updates to the zone resource records in the domain and reverse files. This is shown in the following example:

```
test.itsc.raleigh.ibm.com ns-updates.test.itsc.raleigh.ibm.com
Pb7bySIfzXcW...

200.200.200.in-addr.arpa ns-updates.test.itsc.raleigh.ibm.com
KlexSRMP/q/k...
```

3. Start the DDNS server.
4. If you have a DHCP server configured for DDNS updates, you need to add an entry into the DDNS.DAT file using NSUPDATE -g that represents a wildcard entry for the zones that DHCP will update (*.test.itsc.raleigh.ibm.com).

Note: KEY and SIG resource records, as well as encryption keys, always use a single line. The examples were indented for illustration purposes only.

Configuring for Presecured Mode Operation: IBM Open Edition's Dynamic DNS server supports two modes of securing updates to a dynamic DNS zone. In the default mode, called **dynamic secured**, any host that complies with the DDNS protocol can create resource records in a zone declared as dynamic. After created, these records can only be updated by the administrator or by the host that created them.

In presecured mode, only hosts that have been preauthorized by a DDNS administrator may create resources in a particular dynamic zone. After created, these records can only be updated by the administrator or by the host that has been preauthorized.

Functionally, the difference in these modes is whether or not the DDNS server will allow the creation of a KEY RR or whether it must already have a KEY RR defined for a resource in the zone. In the case of presecured mode, the KEY RR must be already defined in the zone before an update is accepted. Specifically, the administrator must enter the KEY RR data in the domain file for each client that will be making updates.

To configure a particular DNS dynamic zone for presecured mode operation, you must:

1. Specify **dynamic presecured** on the primary zone statement in the DDNS server boot file and create the corresponding domain file.
2. Run the nsupdate -g command to create an /etc/ddns.dat with the zone key information. Create a zone KEY RR.
3. Start the DDNS server by entering **named** at an z/OS UNIX command prompt.
4. For each client host:
   a. Use **nsupdate** with the -g parameter, which will generate a key for the host and save it in /etc/ddns.dat.
   b. Use **nsupdate** with the -a parameter to dynamically register and save a host's KEY RR in the DDNS server domain file.
   c. Manually extract the /etc/ddns.dat file key entry for the host into a separate DDNS.DAT file and distribute it to the end user for installation into their /etc subdirectory.

The following example is a scenario demonstrating an administrator using **nsupdate** to preregister an end user in a **dynamic presecured** domain. Administrator input is highlighted in bold:

```
[C:\]nsupdate -g -h newuser.dynozone.sandbox -p netadmin.dynozone.sandbox
--- NSUPDATE Utility ---
---
```

```
Key Gen ...... succeeded ...


[C:\]nsupdate -a -h newuser.dynozone.sandbox -p netadmin.dynozone.sandbox
--- NSUPDATE Utility ---

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> a
..rrtype (A,PTR,CNAME,MX,KEY,HINFO): key
DDNSUpdate_KEY (Add Flags 0000 Protocol 0 Algid 1
      Keylen 64 Key10-15Õ: AQO+w3f2H1F0zM2Q ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> s
..sig Expiration (secs from now, ENTER for 3600):
..sig KEY pad (ENTER for default of 3110400):
DDNSSignUpdate ...succeeded
DDNSFinalizeUpdate ...succeeded
DDNSSendUpdate ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> q

[C:\]
```

Chapter 9. Policy-Based Networking

Overview

Applications and users of TCP/IP networks have different requirements for the service they receive from those networks. A network that treats all traffic as best effort does not meet the needs of such users. *Service differentiation* is a mechanism to provide different service levels to different traffic types based on their requirements and importance in an enterprise network. For example, it might be critical to provide Enterprise Resource Planning (ERP) traffic better service during peak hours than that of FTP or web traffic. The overall service provided to applications or users, in terms of elements such as throughput and delay, is termed *Quality of Service*(QoS). Network service providers that need to provide different QoS levels express their business goals in *Service Level Agreements* (SLAs). There are two types of service in TCP/IP networks that relate to QoS. The first is *Differentiated Services*, which provides QoS to broad classes of traffic or users, for example all outbound web traffic accessed by a particular subnet. The second is *Integrated Services*, which provides end-to-end QoS to an application, by reserving resources along a data path. For z/OS CS, Integrated Services is largely provided by the RSVP Agent, which implements the Resource ReserVation Protocol.

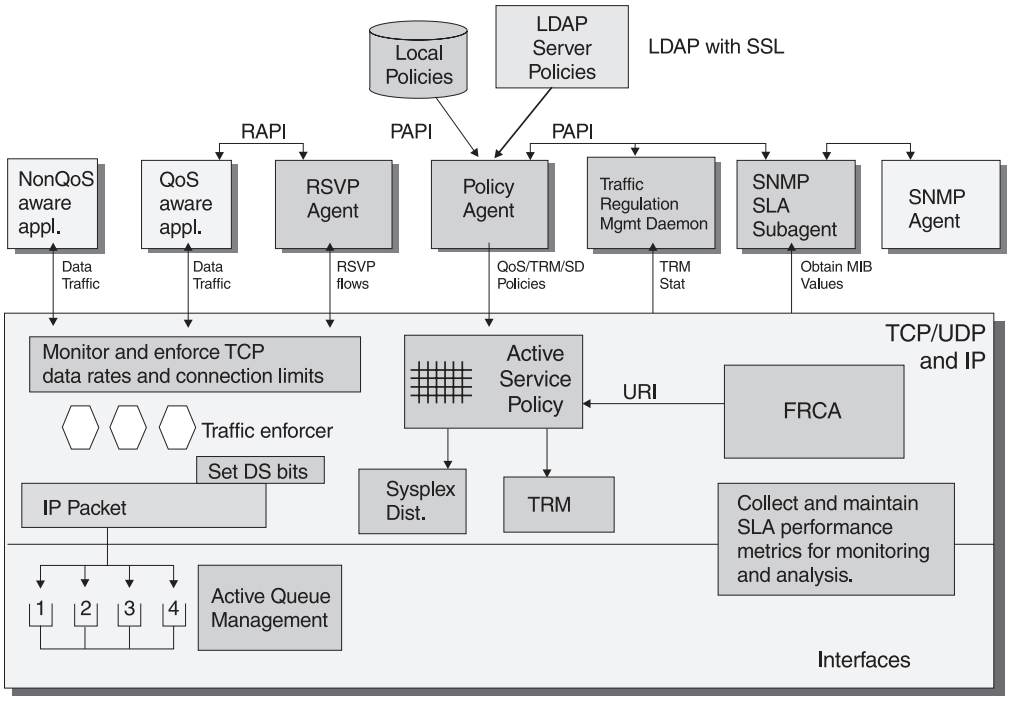


Figure 51. QOS Components in z/OS CS

In addition to QoS levels, there are other types of controls needed for TCP/IP networks, such as those dealing with security or resource balancing. The implementation of SLAs and other network controls on network hosts and routers is provided by *policies*. Policies are an administrative means to define controls for a network, in order to achieve the QoS levels promised by SLAs, or to implement security or resource balancing decisions. To be effective, especially for QoS, consistent mechanisms need to be used throughout the network to classify and differentiate traffic, and to provide a consistent implementation of policy decisions. To achieve this, policies are usually defined in a central policy repository, accessed

by all hosts and routers that need to make policy decisions (*Policy Decision Point* or PDP), or implement such decisions (*Policy Enforcement Point* or PEP).

It is important to be able to monitor the performance of policies as implemented by the network. This leads to the need to define service level management information kept by the PDP and PEP, which can then be analyzed for a variety of network services, from traffic trend analysis to network capacity planning and dynamic QoS level tuning.

Together, service differentiation, policies, and service level performance monitoring form an integral part of the SLA function that is becoming more and more important to network administrators as a means of controlling network traffic flows.

Network administrators can use the OS/390 UNIX Policy Agent (PAGENT) to define policies for their users.

The policies supported by the Policy Agent can be used for any of the following purposes:

- Differentiated Services function
- Integrated Services function
- Sysplex Distributor function
- Traffic Regulation and Management function

In addition, the Policy Agent supports functions other than reading/installing policies, such as Sysplex Distributor policy performance monitoring, and mapping Type of Service (TOS) byte values to outbound interface priorities. All of these functions are further described below.

PAGENT runs in the OS/390 UNIX environment and reads policy definitions from a local configuration file and/or a central repository that uses the Lightweight Directory Access Protocol (LDAP). PAGENT also installs them in one or more z/OS CS stacks. It can be used to replace existing policies or update them as necessary.

Notes:

1. Policies are supported by the stack as an end host system only. When the stack is routing packets, policies are not accessed or applied.
2. Policies defined on an LDAP server use the configuration files and mechanisms provided by the LDAP server product. The definition of the elements of policies is known as the schema. z/OS CS provides the schema definition for policies that may be defined on an LDAP server in two sample files (see below for the names of these samples). These sample files must be installed on the LDAP server as the schema definition. PAGENT uses the OS/390 LDAP Client library to communicate with an LDAP server. Refer to *z/OS SecureWay Security Server LDAP Server Administration and Use* for more information about LDAP and *z/OS SecureWay Security Server LDAP Client Programming* for more information about the OS/390 LDAP Client support.

The RSVP Agent provides Integrated Services functions, such as communicating with RSVP agents on other hosts/routers and reserving resources on certain types of outbound interfaces. The RSVP Agent queries the Policy Agent for policies that relate to RSVP processing.

The SLA subagent allows network administrators to retrieve data and determine if the current set of SLA policy definitions are performing as needed or if adjustments

need to be made. The SLA subagent supports the Service Level Agreement Performance Monitor (SLAPM) MIB. Refer to RFC 2758 for more information about the SLAPM MIB.

Traffic Regulation Management (TRM) support is available to regulate the number of TCP/IP connections to any given port, depending on which hosts are connecting. This is accomplished through TR policy, which is also controlled by the Policy Agent. TCP flooding is a type of denial of service attack, and it is very difficult to detect or distinguish from legitimate heavy traffic. TRM is available to define the maximum number of TCP connections to any given port, and thus regulate the number of connections from a single host based on the status of the port.

A set of sample files is shipped with z/OS CS that provides several functions. First, two samples provide examples of policy definitions and offer additional explanations. Second, two samples are actually the definition of the schema that must be installed on an LDAP server. The last two samples contain the text of draft documents used to develop the Version 2 schema. The sample files are:

/usr/lpp/tcpip/samples/pagent.conf
: This file contains overall policy definition rules, syntax and semantics for defining policies in a configuration file, and examples of such policy definitions.

/usr/lpp/tcpip/samples/pagent.ldif
: This file contains examples of defining policies on an LDAP server.

/usr/lpp/tcpip/samples/pagent_oc.conf
: This file contains the schema object class definitions, and must be included in the LDAP server configuration

/usr/lpp/tcpip/samples/pagent_at.conf
: This file contains the schema attribute definitions, and must be included in the LDAP server configuration.

/usr/lpp/tcpip/samples/pagent_core.txt
: This file contains the draft version of the proposed Core Schema Internet Draft used in CSV2R10.

  Note: This file is provided as-is and there are some differences between the draft and the implementation. The intent of this file is to provide background information on the level of the supported schema.

/usr/lpp/tcpip/samples/pagent_cond.txt
: This file contains the draft version of the proposed Policy Conditions Internet Draft used in CSV2R10.

  Note: This file is provided as-is and there are some differences between the draft and the implementation. The intent of this file is to provide background information on the level of the supported schema.

Note: This documentation refers to both Version 1 and Version 2 when defining policies. Version 1 refers to policy definitions used prior to OS/390 V2R10, while Version 2 refers to current policy definitions. Version 1 provides only a subset of the functionality of Version 2, and some of the semantics are different between the two versions. When defining policies, refer to *z/OS Communications Server: IP Configuration Reference* for more information about V1 and V2 statements.

What Kind of Policy Do You Want?

The policy agent (PAGENT) supports the following types of policies:

- Differentiated Services (DS) policies
- Integrated Services (RSVP) policies
- Traffic Regulation Management (TRM) policies
- Sysplex Distributor (SD) policies

The Policy Agent manages policies. Policies consist of policy rules and policy actions. These are related in a way that is analagous to an 'IF' statement in a program. For example:

```
IF condition THEN action
```

The policy rule serves as the condition in this analogy, while the policy action serves as the action to be performed. Policy rules consist of a variety of selection criteria that act as traffic filters. Traffic can be filtered based on source/destination IP addresses, source/destination ports, protocol, inbound/outbound interfaces, application name, or application specific data. Only packets that match the filter criteria are selected to receive the accompanying action. Policy rules can refer to several policy actions, but only one policy action is executed per policy scope. A given policy action may be referred to by several policy rules. In addition to filter selection criteria, policy rules contain time related information that indicates when the policy rule should be considered active or inactive. Active policy rules are installed in the TCP/IP stack, so they can be applied as traffic filters. Inactive policy rules exist only in the Policy Agent.

The type of policy defined is in general controlled by the policy scope value defined for the policy action. SD policies are an exception. SD policies are a subset of DS policies, so use the DS scope.

Differentiated Services (DS) Policies

Policies for Differentiated Services are used to select and control DS traffic, such as FTP and internet server traffic. Several aspects of connection and throughput control can be specified with these policies, including the following:

- TCP connection limits
- Maximum and minimum TCP connection rates, TCP maximum delay
- Committed access bandwidth (mean rate and peak rate) control/enforcement
- Type of Service (TOS) byte (also known as DS field - 6 bit value) setting, and mapping to S/390 Queued Direct I/O (QDIO) device priorities

The above DS service attributes are enforced by the TCP/IP stack in which the DS policies are installed. For additional information on the enforcement of these attributes, refer to *z/OS Communications Server: IP Configuration Reference*.

Integrated Services (RSVP) Policies

RSVP policies are used to set limits on certain parameters requested by RSVP applications. These applications interact with the RSVP Agent to establish resource reservations along a network path. The reservation requests are in the form of an entity known as a Traffic Specification, or Tspec, which consists of the following values:

- Token bucket mean rate (r)
- Token bucket depth (b)

- Peak rate (p)
- Minimum policed unit (m)
- Maximum packet size (M)

RSVP policies can be used to limit the values requested for (r) and (b), as well as limiting the total number of RSVP reserved flows. The RSVP service attributes are enforced by the RSVP daemon which gets RSVP policies from the Pagent. For additional information on the enforcement, refer to *z/OS Communications Server: IP Configuration Reference* or RFC 1363.

Traffic Regulation Management (TRM) Policies

Policies for Traffic Regulation Management are used to define the maximum number of connections of a TCP port and control the number of connections from a single host to this port. Following are the types of actions under TR policy:

Statistics
: Gather statistics about number of connections and terminations to one or more target ports. These statistics can be analyzed using the trmdstat command. The results of this analysis can be used to establish parameters in the TR policies for TRM traffic regulation control.

Log
: Log events that are out of specification for target ports based on parameters specified in the policies.

Limit
: Perform TCP connection limiting for target ports based on parameters specified in the policies

Sysplex Distributor (SD) Policies

Sysplex Distributor policies are used to specify a set of SD target nodes for a given set of traffic. For example, all traffic destined to a given port/application from a specified subnet can be assigned one group of SD target nodes, while traffic for the same port/application from another subnet can be assigned to a different group of target stacks. These policies be can used in conjunction with other Sysplex Distributor controls to assist in load balancing. For more information, see “Chapter 3. Virtual IP Addressing” on page 109.

The Policy Agent supports all of the above policy types, installing them into one or more TCP/IP stacks as configured. Although RSVP policies are installed into the TCP/IP stack, they are only used for collecting policy statistics. For policy use and limit enforcement, these policies are requested from the Policy Agent by the RSVP Agent, to apply to RSVP reservation requests from RSVP applications.

In addition to supporting the various types of policies, the Policy Agent performs functions related to the Sysplex Distributor . The Policy Agent can be configured to collect network QoS performance data relevant to SD on behalf of policies defined for a target port/application, and assign a *weight fraction* to such policy traffic. This weight is then used by SD - in conjunction with weights assigned by the Workload Manager - to assist in load balancing decisions. This function is performed by the Policy Agent on SD target nodes within the sysplex.

Another function supported by the Policy Agent is to map Type Of Service (TOS) byte values to interface priority values for outbound traffic. The TOS byte is also referred to as the Differentiated Services (DS) byte as an alternative definition (refer to RFC 2474). Note that outbound interface priority values are only meaningful for QDIO interfaces. A set of mappings can be defined to cover various TOS byte

values and map them to an appropriate interface priority. All outbound packets over the associated interfaces with a given TOS byte value will then be assigned the corresponding priority value.

Where Do You Want to Define Your Policies?

Policies can be defined in the Policy Agent Configuration file, in the LDAP server, or both. Policies from both sources are combined into a single list. Note that this requires unique policy object names. For policies defined on the LDAP server, the Distinguished Name (DN) must be unique, but the user-friendly name does not have to be unique (although it is recommended). The Policy Agent appends a unique suffix if required to make LDAP user-friendly names unique within the scope of policies defined on the LDAP server. When policies from a configuration file are combined with LDAP defined policies, the LDAP user-friendly names must be unique with respect to the names defined in the configuration file. Any policy objects with duplicate names at this point are discarded by the Policy Agent. The following table shows some of the advantages and disadvantages of both.

Type	Advantages	Disadvantages
Configuration File	• Policies are easy to define and change. • Definitions are local - no connection needed. • Very little overhead when policies do not change. • Scope TR policies can only be defined in a configuration file.	• Full power of policy definitions not available (for example can only define simple rules). • Each host must maintain its own policy definitions.
LDAP Server	• Central policy definitions for many hosts. • Policy definitions can be shared/reused between different platforms/hosts. • Full power of policy definitions available (for example can define complex rules). • Allow roburst hierarchical policy definition (Sysplex, LPAR, TCP/IP image).	• Policy definitions are more complex. • LDAP server must be queried at each refresh interval to check for new/changed policies. • Scope TR policies cannot be defined on an LDAP server.

Policy Agent (PAGENT)

Configuring the Policy Agent (PAGENT)

The Policy Agent (PAGENT) is responsible for reading policies from a configuration file and/or an LDAP server. Before defining policies, some basic operational characteristics of PAGENT need to be configured. Follow these steps to configure these items.

1. Define the TcpImage statement(s) in the main PAGENT configuration file.

   The Policy Agent can be configured to install policies on one or more TCP/IP stacks, or images. Each TCP/IP stack is configured using a TcpImage statement in the main configuration file. A secondary configuration file can be defined for

any given stack, a set of stacks can share configuration information in the main configuration file, or a combination of these techniques can be used.

To install different sets of policies to different stacks, configure each image with a different secondary configuration file. In this case, each image can be configured with a different policy refresh interval if desired. The refresh interval used for the main configuration file will be the smallest of the values specified for the different stacks.

Note: When the main configuration file is an MVS data set, it is reread at each refresh interval (which is the smallest of the individual stack refresh intervals), regardless of whether it has actually been changed or not. Because PAGENT restarts all stack-related processing when the main configuration file is reread, this effectively makes the refresh interval for all stacks the same as this smallest configured interval.

To install a common set of policies to a set of stacks, don't specify secondary configuration files for each image.

In this case, there is only one configuration file (the main one) and the policy information contained in it is installed to all of the configured stacks. Different refresh intervals can also be configured for each image, but would probably be less useful in this case.

In either case, it is possible that TCP/IP stacks configured to the Policy Agent are not started or even defined. The Policy Agent will fail when trying to connect to those stacks and log appropriate error messages.

The Policy Agent does not end when any (or all) stacks end. When the stack(s) are restarted, active policies are automatically reinstalled.

The TcpImage statement specifies a TCP/IP image and its associated configuration file to be installed to that image. The following example installs the policy control file */tmp/TCPCS.policy* to the *TCPCS* TCP/IP image, after flushing the existing policy control data:

```
TcpImage TCPCS /tmp/TCPCS.policy FLUSH
```

2. Define the appropriate logging level.

   The LogLevel statement is used to define the amount of information to be logged by the Policy Agent. The default is to log only error, console, and warning messages. This might be appropriate for a stable policy configuration, but more information might be required to understand policy processing or debug problems when first setting up policies or when making significant changes. Specify the LogLevel statement with the appropriate logging level in the main configuration file.

   Note: The maximum logging level (511) can produce a significant amount of output, especially with large LDAP configurations. This is not a concern if an HFS log file is used, because PAGENT uses a set of log files with a finite size in a round-robin configuration (the number and size of these files is controllable with the PAGENT_LOG_FILE_CONTROL environment variable). But when using the syslog daemon as the log file, the amount of log output produced should be taken into consideration.

3. Define security product authorization for the Policy Agent.

Because the Policy Agent can affect system operation significantly, security product authority (for example, RACF) is required to start the policy agent. Refer to the EZARACF sample in SEZAINST for sample commands needed to create the profile name and permit users to it.

Defining Policies in a Policy Agent Configuration File

Configure the following statements in the configuration file to define policies:

- PolicyAction
- PolicyRule

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these statements.

The following sections contain examples of these tasks.

Note: These examples are for illustrative purposes only. The policies deliberately use a wide variety of attributes, and they do not necessarily represent real-world usage.

Defining Differentiated Services Policies Using PAGENT

The goal of this Differentiated Services policy is to map a subset of the traffic outbound from an FTP server.

This policy is identified as a Differentiated Services policy by the PolicyScope DataTraffic attribute on the PolicyAction statement, as well as the use of several DS-only attributes.

These differentiated services rates are defined as follows:

DiffServInProfileRate is the average or mean rate that is desired to be transmitted over time. For example, 256 kilobits per second as in the example in this section.

DiffServInProfileTokenBucket is the ″burst″ size. This is how much data is allowed to be sent from the application to TCP/IP and still be allowed to be transmitted at the mean rate above. It is suggested, if the application is not ″policing″ itself, that this burst size be at least one second's worth of data. Otherwise, if the application is sending large amounts of data at one time to TCP/IP, TCP/IP will slow that application down via congestion windows, and the mean rate may not be achieved.

DiffServInProfilePeakRate is the highest rate that is allowed to be transmitted for a shorter interval of time. For example, though a customer may only want on average 256 Kilobits of data per second, they may allow a peak of 512 kilobits of data for 1/4 second. The peak rate is used to control the spacing of outbound packets on the transmission line. By having a smaller peak rate, there will be longer spacing between packets, and thus less ″burstiness″ of traffic and increased efficiency. Higher peak rates result in shorter spacing and increased ″burstiness,″ which can result in lower link utilization. However, some applications may require it, such as real time or video data.

DiffServInProfileMaxPacketSize is the amount of data that will be policed at the peak cell rate. For example, if the peak rate is 512 Kilobits per second, and the max pkt size is 120 Kilobits, TCP/IP will only allow about 10 packets of size 1492 bytes to be transmitted every .23 seconds. Again, if an application is sending large amounts of data at one time to TCP/IP, TCP/IP will enforce the peak rate, and anytime more than 10 packets are sent within 1/4 second, TCP/IP will begin to slow

this TCP connection. The peak rate can be achieved over a longer period of time if the maxpktsize is entered in larger multiples of packets. However, this will cause greater ″burstiness″ as described above. For example, if the maxpacketsize is entered as 240 kilobits in the example, TCP/IP will allow 20 pkts in a .23 second range before enforcing slowdown.

Note that the peak rate cannot be enforced without mean rate policing. However, you can enforce mean rate without peak rate. Also, setting of these parameters depends on the type of applications and the network that carries it.

The following statements apply to the example in this section:

- The policy rule selects traffic to the 211.40.100 subnet (assuming a subnet mask of 255.255.255.0), to any destination port in the range 5000 to 9999, outbound on interface 9.67.116.98, orginated by ports in the range 20-21 (FTP outbound data connection uses port 20) with an application name that starts with ″FTP″.
- The policy rule is active for the first 10 days of each month, on weekdays between 6 a.m. and 8 p.m. local time.
- The policy action specifies that the TOS byte be set to ’01000000’ for traffic that conforms to this policy.
- The action also establishes a TCP round trip time of 50 milliseconds, to be used for performance monitoring using the SLA subagent.
- The action also limits the number of connections that map to this policy to 100.
- The action establishes a *token bucket* traffic conditioner with a mean rate of 128 kilobits per second and a burst size of 4 kilobytes. Any traffic that exceeds these specifications will be sent as *best effort*, with an accompanying TOS byte of ’00000000’.

```
TcpImage  TCPCS FLUSH 600
LogLevel 511
PolicyRule                              DiffServ_Rule1
{
   OutboundInterface                    9.67.116.98
   DestinationAddressRange              211.40.100.0-211.40.100.255
   DestinationPortRange                 5000:9999
   SourcePortRange                      20-21
   PolicyActionReference                DiffServ_Action1
   ApplicationName                      FTP*
   DayOfMonthMask                       11111111110000000000000000000000
   DayOfWeekMask                        0111110
   TimeOfDayRange                       06:00-20:00
}
PolicyAction                            DiffServ_Action1
{
   PolicyScope                          DataTraffic
   OutgoingTOS                          01000000
   MaxDelay                             50
   MaxConnections                       100
   Permission                           Allowed
   DiffServInProfileRate                256       # 256 Kbps
   DiffServInProfileTokenBucket         512       # 512 Kbits
   DiffServInProfilePeakRate            512       # 512 Kbits
   DiffServInProfileMaxPacketSize       120       # 120 Kbits
   DiffServOutProfileTransmittedTOSByte 00000000
   DiffServExcessTrafficTreatment       BestEffort
}
```

Defining RSVP Policies Using PAGENT

The goal of this RSVP policy is to establish limits on resource reservations requested by RSVP applications using the RSVP API (RAPI) interface. The policy is

identified as an RSVP policy by the PolicyScope attribute on the PolicyAction statement, as well as the use of RSVP-only attributes.

The following statements apply to the example in this section:

- The policy rule selects traffic from source ports in the range 8000 to 8001, with a protocol ID of 6 (TCP).
- The policy action specifies that the TOS byte be set to '01100000' for traffic that conforms to this policy.
- The action limits the type of RSVP service requested by RSVP applications to Controlled Load. Applications requesting Guaranteed service are ″downgraded″ to using Controlled Load service. In addition, the action limits the mean rate and token bucket size to 50000 bytes per second and 6000 bytes, respectively. These values are requested by RSVP applications in the traffic specification, or Tspec.
- The action also limits the number of active RSVP flows that map to this policy to 10.

```
TcpImage  TCPCS FLUSH 600
LogLevel 511
PolicyRule                              RSVP_Rule1
{
   SourcePortRange                      8000 8001
   ProtocolNumberRange                  6
   PolicyActionReference                RSVP_Action1
}
PolicyAction                            RSVP_Action1
{
   PolicyScope                          RSVP
   OutgoingTOS                          01100000
   FlowServiceType                      ControlledLoad
   MaxRatePerFlow                       400    # 50000 bytes/second
   MaxTokenBucketPerFlow                48     # 6000  bytes
   MaxFlows                             10
}
```

Defining TRM Policies Using PAGENT

The goal of these TR policies are to demonstrate the different phases of TR processing, as detailed below. The policies are identified as TR policies by the PolicyScope TR attribute on the PolicyAction statement, as well as the use of TR-only attributes.

The following example contains three policies for the 3 *phases* of TR processing. In order to determine the values of the key TR parameters (TotalConnections and Percentage), a three step TR process can be conducted.

Phase 1: The first policy is to determine the maximum number of concurrent connections and the maximum number of concurrent connections held by a single host on port 23 during one or more two hour intervals. A suggested TotalConnections and Percentage will be determined by TR processing for each two hour interval. Statistics data should be gathered over multiple intervals. Based on this data, a user can determine what might be best values for TotalConnections and Percentage.

The following statements apply to the example in this section.

- Only TypeAction Statistics is defined.
- The policy rule gathers information on *normal traffic* for a port.
- This policy will give a statistics report every two hours, with a suggested limit and percentage for that interval.

```
PolicyAction                            telnetAction
{
   PolicyScope                          TR
   TypeActions                          Statistics
   TimeInterval                         120
   LoggingLevel                         5
}

PolicyRule                              telnetRule
{
   DestinationPortRange                 23
   PolicyActionReference                telnetAction
}
```

Phase 2: The values of TR parameters TotalConnections and Percentage, determined during the phase 1, can be tried out by adding them to the policy, and adding Log to the TypeActions parameter. This will log any out of specification connection request without denying the request. With this information, a user can determine whether the policy is correct or not.

The following statements apply to the example in this section.

- TypeActions specifies both Statistics and Log. This allows the current values of TotalConnections and Percentage to be tested while also continuing to gather statistics.
- This is a policy that could be used to try out a connection limit and percentage, but not actually put it into effect.
- This particular configuration might be an internal server, that the customer knows will only need to be used by a few hosts. Thus, the high percentage.
- The policy uses the default LoggingLevel.

```
PolicyAction                            telnetAction
{
   PolicyScope                          TR
   TypeActions                          Statistics Log
   TotalConnections                     1000
   Percentage                           80
}

PolicyRule                              telnetRule
{
   DestinationPortRange                 23
   PolicyActionReference                telnetAction
}
```

Phase 3: After trying out the TR policy and determining that it will not block legitimate connection requests under normal conditions, Limit is added to TypeActions to stop any host from requesting more connections than the TR policy allows.

The following statements apply to the example in this section.

- TypeActions specifies both Log and Limit. This allows the current values of TotalConnections and Percentage to be enforced while also continuing to log the results.
- This is a policy that could be used after testing TotalConnections and Percentage with Log mode (as above).
- The policy is now being enabled with the Limit keyword. This type of configuration may be one used for an internet server that will be used by many different hosts.

```
PolicyAction                            telnetAction
{
   PolicyScope                          TR
   TypeActions                          Log Limit
   TotalConnections                     5000
   Percentage                           5
}

PolicyRule                              telnetRule
{
   DestinationPortRange                 23
   PolicyActionReferenc                 telnetAction
}
```

Note: TotalConnections is not associated with MaxConnections. The former defines total number of connections allowed on a local port, the latter defines the maximum number of connections allowed to a subnet or host. However, when the MaxConnections is applied to a host, and the MaxConnections is larger than the number of connections allowed to a single host under a TR policy, the connection request will be granted to the host.

Defining Sysplex Distributor Policies Using PAGENT

The goal of this Sysplex Distributor policy is to limit the number of SD target stacks for inbound FTP traffic from a given subnet. The policies are identified as SD policies by the presence of the OutboundInterface attribute on the PolicyAction statement.

The following statements apply to the example in this section:

- This policy is defined on the Sysplex Distributor distributing stack.
- The policy rule selects incoming FTP connection requests from subnet 9.37.80.0.
- Incoming connection requests that map to this rule will be distributed to target stacks identified by XCF links 9.67.116.1 and 9.67.116.2, based on Workload Manager (WLM) information and QoS information (if this function is activated on the target stacks).
- If neither of these target stacks is available to service incoming requests (either the node is down or the FTP server is not active), then Sysplex Distributor will distribute the requests to any available target stack.

  Note: If the OutboundInterface 0.0.0.0 statement were not present, and neither of the defined target stacks were available, Sysplex Distributor would reject the request.

```
policyAction        ftpaction
{
    MaxConnections 50        # Limit FTP concurrent connections to 50.
    MaxRate        400       # Limit FTP connection throughput to 400 Kbps.
    OutgoingTOS    01000000  # the TOS value of outgoing FTP packets.
    outboundinterface  9.67.116.1
    outboundinterface  9.67.116.2
    outboundinterface  0.0.0.0
}

policyRule           ftprule
{
    ProtocolNumberRange 6
    DestinationPortRange         20 21
    SourceAddressRange      9.37.80.0 9.37.80.255
    policyactionreference   ftpaction
}
```

Note: If you are using Telnet with multiple stacks in conjunction with the Sysplex Distributor, see “Chapter 6. Accessing Remote Hosts Using Telnet” on page 225 for more information.

Activating the Sysplex Distributor Policy Performance Monitoring Function

The following example illustrates how to activate the policy performance monitoring function for Sysplex Distributor.

Note: This function is activated on SD target stacks and is used to monitor the performance of outbound traffic being serviced by the target stacks. The goal is to detect TCP traffic that exceeds defined thresholds for dropped packets and/or timeouts, and derive a QoS weight fraction for the target stack. This weight fraction is then used to reduce the WLM weight assigned to the target stacks, so that the SD distributing stack can take QoS performance into account.

The following statements apply to the example in this section:

- The policy performance monitoring sampling interval is 60 seconds.
- PAGENT assigns a loss ratio weight fraction of 25% when the TCP loss ratio (dropped packets to total packets) starts to exceed 2%.
- This weight fraction is increased to 50% when the loss ratio starts to exceed 4%, continuing in this manner up to the maximum loss ratio weight fraction of 95%.
- In a similar manner, a TCP timeout weight fraction of 50% is assigned when the timeout ratio starts to exceed 5%, increasing up to a maximum timeout weight fraction of 100%.
- The individual weight fractions are added together to form a single QoS weight fraction for the target stack, up to a maximum weight fraction of 100%.
- The QoS weight fraction is used at the SD distributing stack to reduce the WLM weight. For example, if the WLM weight is 40, a weight fraction of 50% results in the weight being reduced to 20.
- The traffic to be monitored must be represented by at least one Differentiated Services policy defined for the target application (in this example a policy is defined for FTP).

```
PolicyPerfMonitorForSDR  enable
{
   samplinginterval 60
   LossRatioAndWeightFr 20 25
   TimeoutRatioAndWeightFr 50 50
   LossMaxWeightFr 95
   TimeoutMaxWeightFr 100
}
```

Only one policy rule and policy action are defined here. As a result, only FTP QoS performance information is monitored by Pagent for Sysplex Distributor to route incoming FTP connections to this target node relative to other target nodes which presumably can also accept FTP requests.

```
policyAction ftpaction
{
MaxConnections 50 # Limit FTP concurrent connections to 50.
MaxRate 400 # Limit FTP connection throughput to 400 Kbps.
OutgoingTOS 01000000 # the TOS value of outgoing FTP packets.
}

policyRule ftprule
{
```

```
ProtocolNumberRange 6
SourcePortRange 20 21
policyactionreference ftpaction
}
```

Mapping Type of Service (TOS) Values to Interface Priorities Using PAGENT

The following example shows a mapping of various TOS byte values to associated interface priority values. Note that the mapping can be applied to individual interfaces or all interfaces:

- The first example defines a mapping for a specific interface. The second example shows a different mapping for all other interfaces.
- The subnet mask defines the bits in the TOS byte that are significant. These examples use the leftmost 3 bits.
- The set of mappings define the complete set of TOS byte values and the priority to be assigned for each value.

```
SetSubnetPrioTosMask
{
  SubnetAddr         10.10.1.5
  SubnetTosMask      11100000
  PriorityTosMapping 1 11100000
  PriorityTosMapping 1 11000000
  PriorityTosMapping 2 10100000
  PriorityTosMapping 2 10000000
  PriorityTosMapping 2 01100000
  PriorityTosMapping 3 01000000
  PriorityTosMapping 4 00100000
  PriorityTosMapping 4 00000000
}
SetSubnetPrioTosMask
{
  SubnetTosMask      11100000
  PriorityTosMapping 1 11100000
  PriorityTosMapping 1 11000000
  PriorityTosMapping 1 10100000
  PriorityTosMapping 1 10000000
  PriorityTosMapping 2 01100000
  PriorityTosMapping 2 01000000
  PriorityTosMapping 3 00100000
  PriorityTosMapping 4 00000000
}
```

LDAP Server

Lightweight Directory Access Protocol (LDAP) is a fast-growing technology for accessing common directory information. LDAP has been embraced and implemented in most network-oriented middleware. As an open, vendor-neutral standard, LDAP provides an extendable architecture for centralized storage and management of information that needs to be available for today's distributed systems and services.

After a fast start, it can be assumed that LDAP has become the de facto access method for directory information, much the same as the Domain Name System (DNS) is used for IP address look-up on almost any system on an intranet and on the Internet.

Note: If the OS/390 LDAP server is used, a DB2 backend is required.

Defining Policies in an LDAP Server

Refer to SEZAINST for sample configuration files (EZAPAGOC and EZAPAGAT) that you can use on an LDAP server.

Overview of the Object Classes

Policies defined on an LDAP server are comprised of one or more objects, each with a defined object class, and a unique name. Object names are in the form of LDAP Distinguished Names (DNs), which are text strings composed of individual parts known as Relative Distinguished Names (RDNs). The structure of the naming defines a hierarchical tree, in a manner similar to directories in a hierarchical file system. For example, consider the following set of objects:

```
   Object 1, DN: o=IBM, c=US
Object 2, DN: group=group_1, o=IBM, c=US
Object 3, DN: group=group_5, o=IBM, c=US
Object 4, DN: subgroup=group_1_sub_A, group=group_1, o=IBM, c=US
```

This set of objects can be viewed as a tree, with Object 1 as the root. Objects 2 and 3 are branches under the root, with Object 4 a branch under Object 2. The names used are arbitrary, although certain conventions are followed, such as o= for the organization, and c= for the country. Refer to *z/OS SecureWay Security Server LDAP Server Administration and Use* (SC24–5861) for more information on LDAP naming.

Object class names define the type of each LDAP object. The following object classes are recognized by the Policy Agent:

Object Class Name	Purpose of Object
Top	Used to anchor the LDAP hierarchical tree root.
ibm-policyGroup	Defines a policy group object.
ibm-policyGroupContainmentAuxClass	Defines an auxiliary class for binding a policy group object to another policy group.
ibm-policyRuleContainmentAuxClass	Defines an auxiliary class for binding a policy rule object to another policy group.
ibm-policyRule	Defines a policy rule object.
ibm-policyCondition	Defines a policy condition object.
ibm-networkingPolicyCondition	Defines a subclass of ibm-PolicyCondition used to define networking policy conditions.
ibm-routeConditionAuxClass	Defines an auxiliary class to represent routing conditions for a policy rule.
ibm-applicationConditionAuxClass	Defines an auxiliary class to represent application and transport conditions for a policy rule.
ibm-hostConditionAuxClass	Defines an auxiliary class to represent host (end-point) conditions for a policy rule.
ibm-policyTimePeriodCondition	Defines an auxiliary class to represent time periods during which a policy rule is considered to be active.
ibm-policyAction	Defines a policy action object.
ibm-serviceCategories	Defines an auxiliary class to represent a set of QoS attributes for a policy action.

Policy objects are used to accomplish the following objectives:

- Group related objects together. Policy groups can contain related policy rules, and can also contain other policy groups.This allows objects to be grouped in various administrative ways.
- Specify conditions for a policy rule. The conditions are used to filter traffic packets, and can specify attributes such as source and destination port, application name, protocol, etc. Policy rules can be either simple or complex, depending on the nature of the specified conditions. When a single condition is specified, the rule is a simple rule. This single condition can be composed of any of the condition attributes, but only one instance of each. For example, only a single destination port range can be specified in a simple rule.Complex rules specify more than one condition. The specified conditions are organized into one or more levels, and each level is composed of one or more conditions. Each condition can be composed of one instance of any of the condition attributes. The conditions can thus be thought of as a two-dimensional array. Any individual condition can be negated. Two types of processing are applied to the conditions, depending on the specified condition list type:
  - Disjunctive Normal Form (DNF). DNF conditions are logically ANDed at each level, and ORed between levels.
  - Conjunctive Normal Form (CNF). CNF conditions are logically ORed at each level, and ANDed between levels.

  For example, suppose five conditions are specified, two at one level and three at another:

  ```
  Level 1: C1, NOT C2
  Level 2: C3, C4, C5
  ```

  If DNF is specified, the conditions are evaluated as:

  ```
  (C1 AND NOT C2) OR (C3 AND C4 AND C5)
  ```

  CNF evaluation of the same conditions is:

  ```
  (C1 OR NOT C2) AND (C3 OR C4 OR C5)
  ```

  This allows a wide variety of conditional logic to be defined for policy rules.
- Specify time periods during which policy rules are active. Active policy rules are those that are installed in a TCP/IP stack by the Policy Agent. A wide variety of attributes are available to specify time periods, and up to 25 time periods can be specified for any policy rule. The policy rule is active if any of the specified time intervals include the current time.
- Specify actions to take on behalf of traffic that maps to active policy rules, based on the evaluation of its conditions. Actions contain a scope attribute that indicates the type of policy being defined, namely Differentiated Services, RSVP, or both Differentiated Services and RSVP. Up to four actions can be specified for each rule, but only one action per scope can be active at a time.

LDAP objects can refer to other objects, using the DN of the referenced object. For example, a policy rule can be separated from its conditions and time periods, with those objects being referenced by the rule object.

Each LDAP object is composed of a number of attributes. Some of the attributes are generic LDAP attributes that apply to all LDAP objects. Other attributes are used only for Version 1 policy definitions. All other Version 2 policy attributes must begin with a unique prefix:

```
ibm-
```

The design of the LDAP object tree should be carefully thought out. The Policy Agent uses information defined on the ReadFromDirectory statement to perform an initial search on the LDAP server. This initial set of objects is retrieved in one operation. While processing this set of objects, references to other objects may be found, resulting in additional LDAP search operations to retrieve these objects one at a time. It's therefore possible to design an LDAP tree such that a minimal set of objects is initially retrieved, followed by many additional LDAP retrievals. If the total set of objects is large, there is a performance impact to retrieving objects in this manner. If possible, try to design the tree and the ReadFromDirectory parameters to retrieve the largest set of objects initially, to achieve the best performance.

Step 1: Configure PAGENT to use LDAP Server via the ReadFromDirectory Statement

The ReadFromDirectory Statement in the Policy Agent configuration file initializes PAGENT as a LDAP client. The policies are downloaded from the LDAP server, along with the policies specified in the PAGENT configuration file.

When configuring the ReadFromDirectory statement, first specify the name (or IP address) and port of the primary server and the same for the backup server (if one is used).

Note: When using the OS/390 LDAP server, the server listens on a separate port for SSL connections. This means that you should specify the correct port depending on whether or not SSL is used.

Next, configure other connection attributes. The Policy Agent (as an LDAP client) must log in to the LDAP server. The userid and password for logging in must be configured on the ReadFromDirectory statement. The userid is also known as Distinguished Name for userid, and it is in the form of an LDAP DN. If the userid and password are not specified, the Policy Agent uses anonymous login to connect to the server.

The LDAP server might be running either Version 2 or Version 3 of the LDAP protocol, which must also be configured on the ReadFromDirectory statement. This statement also configures the version of the schema to be retrieved from the server.

Finally, configure attributes to indicate how to search the LDAP server for policies. For the Version 1 schema, a base DN to start the search, and a *selector tag* value are configured. The selector tag is used to match against the SelectorTag attribute in the policy objects. For Version 1, the Policy Agent also automatically includes the stack name when searching for policies; this value is matched against the TcpImageName attribute in the policy objects. For the Version 2 schema, a base DN to start searching is also configured. This DN can specify a single LDAP object, a policy group, or an LDAP subtree containing many objects. For filtering the search, two keywords can be configured. One keyword matches against the ibm-policyGroupKeywords attribute in policy group objects, and the other matches against the ibm-policyRuleKeywords attribute in policy rule objects.

The following example does the following:

- Connects to the LDAP server at IP address 9.100.1.1, using the default port 389.
- Specifies a userid and password to log in to the server.
- Specifies LDAP protocol version 3.
- Specifies schema version 2.
- Starts searching at the DN g=policy, o=IBM, c=US object/subtree.

- Only selects policy rules that contain either the ″POLICY″ or ″EASTERN″ keywords.

```
ReadFromDirectory
{
LDAP_Server 9.100.1.1
LDAP_DistinguishedName cn=root, o=IBM, c=US
LDAP_Password 4qr56jb
LDAP_ProtocolVersion 3
LDAP_SchemaVersion 2
SearchPolicyBaseDN g=policy, o=IBM, c=US
SearchPolicyRuleKeyword POLICY
SearchPolicyRuleKeyword EASTERN
}
```

Step 2: Optionally Add SSL to PAGENT Connection to LDAP

The Secure Sockets Layer (SSL) protocol begins with a handshake. During the handshake, the client authenticates the server, the server optionally authenticates the client, and the client and server agree on how to encrypt and decrypt information.

Server Authentication: When using SSL to secure communications, the SSL authentication mechanism known as server authentication is used. With server authentication, the LDAP server must have a digital certificate which authenticates the server to the Pagent client. The server supplies the client with the certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure communication channel is established between the LDAP server and the Pagent client.

For server authentication to work, the LDAP server must have a private key and associated server certificate in the server keyring file.

To conduct commercial business on the Internet, you might use a widely known Certificate Authority (CA) such as VeriSign, to get a high assurance certificate. For a relatively small private network within your own enterprise or group, you can issue your own certificates, called self-signed certificates, for your own use.

Client Authentication: When using SSL Client Authentication, the client passes a digital certificate to the server as part of the SSL handshake. To pass authentication, the Certificate Authority (CA) that signed the client certificate must be considered trusted by the server.

Self-signed Server Certificates: Normally, a server certificate should be obtained from a known CA. However, for testing, an installation might use a self-signed server certificate. Because the clients will not know about the issuer of the self-signed server certificate, in most cases it is necessary to add the server's self-signed certificate to the client's signer certificates.

The GSKKYMAN utility is used to create public/private key pairs and certificate requests, receive certificate requests into a key ring, and manage keys in a keyring. GSKKYMAN is documented in *z/OS System Secure Sockets Layer Programming*. GSKKYMAN is shipped with OS/390 in System SSL which is part of Cryptographic Services Base element of OS/390.

The Pagent connection to LDAP can be secured using SSL by tailoring the following parameters on the ReadFromDirectory statement listed below. This allows for protection of policy retrieval from an LDAP server.

- LDAP_SSLKeyringFile

- LDAP_SSLKeyringPassword
- LDAP_SSLName

For more detail about these parameters, refer to *z/OS Communications Server: IP Configuration Reference*. Additional information about the concepts of cryptography and SSL can be found at the following Web sites:

- http://home.netscape.com/eng/ssl3/
- http://www.verisign.com/repository/crptintr.html

Defining Differentiated Services Policies Using LDAP

The goal of this Differentiated Services Policy is to map a subset of the traffic outbound from an FTP server.

This policy is identified as a Differentiated Services policy by the ibm-PolicyScope attribute in the ibm-PolicyAction object class.

The following statements apply to the example in this section:

- The policy rule selects traffic to the 211.40.100 subnet (assuming a subnet mask of 255.255.255.0), to any destination port in the range 5000 to 9999, outbound on interface 9.67.116.98, orginated by ports in the range 20-21 (FTP outbound data connection uses port 20) with an application name that starts with *FTP*.
- The policy action specifies that the TOS byte be set to '01000000' for traffic that conforms to this policy.
- The action also establishes a TCP round trip time of 50 milliseconds, to be used for performance monitoring using the SLA subagent.
- The action also limits the number of connections that map to this policy to 100.
- The action establishes a *token bucket* traffic conditioner with a mean rate of 128 kilobits per second and a burst size of 4 kilobytes. Any traffic that exceeds these specifications will be sent as *best effort*, with an accompanying TOS byte of '00000000'.

```
dn:o=IBM, c = US
objectclass:top

dn:g=policy, o=IBM, c = US
objectclass:top

dn:pr=DiffServ_Rule1, g=policy, o=IBM, c= US
objectclass:ibm-policyRule
ibm-policyRuleName:DiffServ_Rule1
cn:Differentiated Services Rule 1
ibm-policyRuleEnabled:1
ibm-policyRuleConditionListType:1
ibm-policyRuleConditionList:1:+:pc=cond1, g=policy, o=ibm, c=us
ibm-policyRuleActionList:0:pa=DiffServ_Action1, g=policy, o=ibm, c=US
ibm-policyRuleValidityPeriodList:pc=period1, g=policy, o=ibm, c=us
ibm-policyRuleKeywords:DiffServPolicyRules
ibm-policyRuleSequencedActions:3

dn:pc=period1, g=policy, o=IBM, c= US
objectclass:ibm-policyTimePeriodCondition
ibm-policyConditionName:timeperiod1
cn:active time period 1
ibm-ptpConditionDayOfMonthMask:11111111110000000000000000000000
ibm-ptpConditionDayOfWeekMask:0111110
ibm-ptpConditionTimeOfDayMask:060000:200000
description:time period 1

dn:pc=cond1, g=policy, o=IBM, c= US
objectclass:ibm-networkingPolicyCondition
```

```
objectclass:ibm-routeConditionAuxClass
objectclass:ibm-hostConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
ibm-policyConditionName:condition1
cn:condition 1
ibm-Interface:1--9.67.116.98
ibm-DestinationIPAddressRange:3-211.40.100.0-211.40.100.255
ibm-DestinationPortRange:5000-9999
ibm-SourcePortRange:20-21
ibm-ApplicationName:FTP*

dn: pa=DiffServ_Action1, g=policy, o=ibm, c=US
objectclass:ibm-policyAction
objectclass:ibm-serviceCategories
ibm-policyActionName:DiffServ_Action1
cn:Differentiated Services Action 1
ibm-PolicyScope:DataTraffic
ibm-Permission:allowed
ibm-MaxDelay:50
ibm-MaxConnections:100
ibm-DiffServInProfileRate:128
ibm-DiffServInProfileTokenBucket:32
ibm-DiffServOutProfileTransmittedTOSByte:00000000
ibm-DiffServExcessTrafficTreatment:besteffort
ibm-OutgoingTOS:01000000
```

Defining RSVP Policies Using LDAP

The goal of this RSVP policy is to establish limits on resource reservations requested by RSVP applications using the RSVP API (RAPI) interface. The policy is identified as an RSVP policy by the ibm-PolicyScope attribute on the ibm-PolicyAction object class.

The following statements apply to the example in this section:

- The policy rule selects traffic from source ports in the range 8000 to 8001, with a protocol ID of 6 (TCP).
- The policy action specifies that the TOS byte be set to '01100000' for traffic that conforms to this policy.
- The action limits the type of RSVP service requested by RSVP applications to Controlled Load. Applications requesting Guaranteed service are *downgraded* to using Controlled Load service. In addition, the action limits the mean rate and token bucket size to 50000 bytes per second and 6000 bytes, respectively. These values are requested by RSVP applications in the traffic specification, or Tspec.
- The action also limits the number of active RSVP flows that map to this policy to 10.

```
dn:o=IBM, c = US
objectclass:top

dn:g=policy, o=IBM, c = US
objectclass:top

dn:pr=RSVP_Rule1, g=policy, o=IBM, c= US
objectclass:ibm-policyRule
ibm-policyRuleName:RSVP_Rule1
cn:RSVP Rule 1
ibm-policyRuleEnabled:1
ibm-policyRuleConditionListType:1
ibm-policyRuleConditionList:1:+:pc=cond1, g=policy, o=ibm, c=us
ibm-policyRuleActionList:0:pa=RSVP_Action1, g=policy, o=ibm, c=US
ibm-policyRuleKeywords:RSVPPolicyRules
ibm-policyRuleSequencedActions:3
```

```
dn:pc=cond1, g=policy, o=IBM, c= US
objectclass:ibm-networkingPolicyCondition
objectclass:ibm-applicationConditionAuxClass
ibm-policyConditionName:condition1
cn:condition 1
ibm-SourcePortRange:8000:8001
ibm-ProtocolNumberRange:6

dn: pa=RSVP_Action1, g=policy, o=ibm, c=US
objectclass:ibm-policyAction
objectclass:ibm-serviceCategories
ibm-policyActionName:RSVP_Action1
cn:RSVP Action 1
ibm-PolicyScope:RSVP
ibm-OutgoingTOS:01100000
ibm-FlowServiceType:ControlledLoad
ibm-MaxRatePerFlow:400
ibm-MaxTokenBucketPerFlow:48
ibm-MaxFlows:10
```

Defining Sysplex Distributor Routing Policies Using LDAP

The goal of this Sysplex Distributor policy is to limit the number of SD target stacks for inbound FTP traffic from a given subnet. The policies are identified as SD policies by the presence of the ibm-Interface attribute in the ibm-PolicyAction object class.

The following statements apply to the example in this section:

- This policy is defined on the Sysplex Distributor distributing stack.
- The policy rule selects incoming FTP connection requests from subnet 9.37.80.0.
- Incoming connection requests that map to this rule will be distributed to target stacks identified by XCF links 9.67.116.1 and 9.67.116.2, based on Workload Manager (WLM) information and QoS information (if this function is activated on the target stacks).
- If neither of these target stacks is available to service incoming requests (either the node is down or the FTP server is not active), then Sysplex Distributor will distribute the requests to any available target stack.

  Note: If the ibm-Interface:1–0.0.0.0 attribute were not present, and neither of the defined target stacks were available, Sysplex Distributor would reject the request.

```
dn:o=IBM, c = US
objectclass:top

dn:g=policy, o=IBM, c = US
objectclass:top

dn: pg=groupA, g=policy, o=IBM, c= US
objectclass:ibm-policyGroup
ibm-policyGroupName:groupA
ibm-policyGroupsAuxContainedSet:pg=groupA-1, g=policy, o=ibm, c=US
description:policy group A

dn: pg=groupA-1, g=policy, o=IBM, c= US
objectclass:ibm-policyGroup
ibm-policyGroupName:groupA-1
ibm-policyRulesAuxContainedSet:pr=ftprule, pg=groupA-1, g=policy, o=ibm, c=US
description:policy group A-1

dn:pr=ftprule, pg=groupA-1, g=policy, o=IBM, c= US
objectclass:ibm-policyRule
ibm-policyRuleName:ftprule
```

```
cn:ftp application - rule
ibm-policyRuleEnabled:1
ibm-policyRuleConditionList:1:+:pc=ftpcond, pg=groupA-1, g=policy, o=ibm, c=us
ibm-policyRuleActionList:1:pa=ftpaction, pg=groupA-1, g=policy, o=ibm, c=US
ibm-policyRuleValidityPeriodList:pc=period1, pg=groupA-1, g=policy, o=ibm, c=us
ibm-policyRuleKeywords:testingPolicyRules
ibm-policyRulePriority:2
ibm-policyRuleMandatory:TRUE
ibm-policyRuleSequencedActions:1

dn: pa=ftpaction, pg=groupA-1, g=policy, o=ibm, c=US
objectclass:ibm-policyAction
objectclass:ibm-serviceCategories
ibm-policyActionName:ftpaction
cn:ftp-cos-1
ibm-PolicyScope:DataTraffic
ibm-MaxRate:400
ibm-MaxConnections:50
ibm-OutgoingTOS:01000000
ibm-interface:1--9.67.116.1
ibm-interface:1--9.67.116.2
ibm-interface:1--0.0.0.0

dn:pc=period1, pg=groupA-1, g=policy, o=IBM, c= US
objectclass:ibm-policyTimePeriodCondition
ibm-policyConditionName:timeperiod1
cn:active time period 1
ibm-ptpConditionTime:19990713000000:20021030200000
ibm-ptpConditionMonthOfYearMask:111100111100
ibm-ptpConditionDayOfMonthMask:11111111111111111111111111111111
ibm-ptpConditionDayOfWeekMask:1111111
ibm-ptpConditionTimeOfDayMask:020000:200000
ibm-ptpConditionTimeZone:Z
description:time period 1

dn:pc=ftpcond, pg=groupA-1, g=policy, o=IBM, c= US
objectclass:ibm-networkingPolicyCondition
objectclass:ibm-hostConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
ibm-policyConditionName:hostftpapplcondition1
cn:ftp host and application condition 1
ibm-ProtocolNumberRange:6
ibm-SourceIPAddressRange:2-9.37.80.0-24
ibm-DestinationPortRange:20:21
```

Starting and Stopping PAGENT

You can start the policy agent from the OS/390 shell or as a started task.

Although the /etc/pagent.conf is the default configuration file, a specific search order is used when starting the policy agent. The following order is used:

1. File or data set specified with the -c startup option
2. File or data set specified with the PAGENT_CONFIG_FILE environment variable
3. /etc/pagent.conf
4. hlq.PAGENT.CONF

At initialization, PAGENT creates an HFS file called /tmp/*tcpname*.Pagent.tmp. This occurs for every TCP/IP stack defined on a TcpImage statement.

In this HFS file, tcpname is the name of a TCP/IP stack from a TcpImage statement. During TCP/IP stack initialization, the TCP/IP stack will attempt to modify

a file by this name to notify the PAGENT that the stack has been reactivated. This causes the PAGENT to automatically attempt to reinstall the existing policies to this stack.

To ensure that only one policy agent is started, the policy agent uses the following enqueue:

- Enqueue name is TCP_TCPI
- Resource name is TCPIP.PAGENT

When starting from the shell, note that the pagent executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your PATH statement contains either /usr/sbin or /usr/lpp/tcpip/sbin.

For example, the following command starts PAGENT with these characteristics:

```
pagent -c /u/user10/pldap.conf -l SYSLOGD &
```

- PAGENT uses the configuration file /u/user10/pldap.conf
- PAGENT logs output to the syslog daemon (SYSLOGD) - note that ″SYSLOGD″ must be specified in upper case to obtain this behavior

Use the S PAGENT command on an MVS console or SDSF to start pagent as a started task. A sample procedure is shipped in member EZAPAGSP in SEZAINST.

You can stop PAGENT by:

- Using the cancel command (C PAGENT)
- Using the kill command in the OS/390 shell
- Using the operator command STOP

The following kill command with the TERM signal will enable pagent to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where pid is the pagent process ID.

The pagent process id can be obtained using the following OS/390 UNIX command:

```
ps -A
```

It can also be obtained from the /tmp/pagent.pid file. The /tmp/pagent/pid file is a temporary file created by the policy agent. It contains the process id of the current (or last) invocation of the policy agent.

RSVP

Resource ReSerVation Protocol (RSVP) is a protocol that provides a mechanism to reserve resources in support of Integrated Services. The OS/390 UNIX RSVP agent provides the following services on behalf of applications that want to use Integrated Services:

- An RSVP API (RAPI) that allows applications to explicitly request RSVP services. Using RAPI, applications indicate their intent to send or receive data, describe the characteristics of the data traffic and request that RSVP reserve resources along the data path to provide a given QoS to one or more traffic flows. For more information about RAPI, refer to *z/OS Communications Server: IP Programmer's Reference*.
- Mapping of IP TOS settings to RSVP traffic, using policies defined for RSVP.

- Establishment of resource reservations on ATM interfaces by use of reserved SVC connections.

  Note: Resource reservations cannot be made on interfaces other than ATM for outbound traffic on OS/390. However, RSVP-capable routers in the network can still reserve resources, and the TOS byte can be set for RSVP traffic to provide further means of prioritizing traffic.
- Support for VIPA addresses as well as real IP addresses.
- Communication with other RSVP agents on hosts/routers in the network to communicate application resource reservation requests.

Network administrators can use the OS/390 UNIX Policy Agent to define RSVP-specific policies. These policies can be used to limit the parameters of application-requested resource reservations, provide TOS mappings for RSVP traffic, and limit the number of traffic flows that can use RSVP services simultaneously.

RSVP is designed to be implemented on both end systems (hosts) and routers. Different functions are provided by RSVP in these two environments. The OS/390 RSVP agent is supported as a host RSVP implementation only. It can communicate with router RSVP implementations, but is not itself supported as such. For more information about RSVP, refer to RFC 2205.

Configuring the RSVP Agent

To configure the RSVP agent, update the configuration file to specify RSVP agent operational parameters using the LogLevel, TcpImage, Interface and RSVP statements. Refer to *z/OS Communications Server: IP Configuration Reference* for detailed information about the statements.

To configure the RSVP agent, you must first authorize the RSVP Agent using the security product. See SEZAINST(EZARACF) for SAF considerations for started tasks.

The following is an example of an RSVP configuration file.

This example:

- Runs the RSVP Agent on the stack selected using the standard resolver search order, because a TcpImage statement is not configured.
- Disables RSVP processing on interface 10.11.12.13, while enabling it for all other interfaces.
- Disables traffic control on interface 200.1.1.1. This means that no reservations will be made on this interface.
- Allows a maximum of 50 active RSVP flows per interface.

```
Interface 10.11.12.13 Disabled
{}
Interface 200.1.1.1 Enabled
{
TrafficControl Disabled
}
Interface Others Enabled
{}
Rsvp All Enabled
{
MaxFlows 50
}
```

Starting and Stopping RSVP

RSVP can be started from the OS/390 shell or as a started task.

The RSVP agent uses the following search order to locate the configuration file (highest priority is listed first):

- HFS file or MVS data set specified by the -c startup option. The syntax for an HFS file is '/dir/file', and the syntax for an MVS data set is ″//'MVS.DATASET.NAME'″.
- HFS file or MVS data set specified with the RSVPD_CONFIG_FILE enviroment variable.
- /etc/rsvpd.conf HFS file.
- 'hlq.RSVPD.CONF' MVS data set.

Note: If this file is not present, RSVP is enabled on all network interfaces with default parameters.

When starting from the shell, note that the RSVP executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your path statement (in the profile) contains either /usr/sbin or /usr/lpp/tcpip/sbin.

Use the S RSVPD command on an MVS console or SDSF to start RSVP as a started task. A sample procedure is shipped in member EZARSVPP in SEZAINST.

RSVP can be stopped using the cancel command (C RSVPD) or using the kill command in the OS/390 shell. The following kill command with the TERM signal will enable the RSVP to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where *pid* is the RSVP process ID.

The RSVP process ID can be obtained using the following OS/390 UNIX command:

```
ps  -A
```

It can also be obtained from the /tmp/rsvpd.pid.imagename file. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Service Level Agreement Performance Monitor MIB Subagent

The SNMP Service Level Agreement (SLA) Performance Monitor MIB subagent provides information about defined service policies and performance data for applications which are mapped to those policies. Statistics are retrieved by this subagent and monitored for possible SLA performance deviations. Refer to RFC 2758 for more information about the SLAPM MIB.

Starting and Stopping the SLA Subagent

The SLA subagent can be started from the OS/390 shell or as a started task.

When starting from the shell, note that the SLA subagent executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your PATH statement (in the profile) contains either /usr/sbin or /usr/lpp/tcpip/sbin.

For example, the following command starts the SLA subagent with these characteristics:

```
pagtsnmp -d 1 -t 1800 -c special -P 5000
```

- To connect to the SNMP Agent, a community name of ″special″ and a port of 5000 are used.
- The debugging level is set to 1, meaning internal debugging messages are written to syslogd.
- The MIB table cache time is set to 30 minutes.

Use the S PAGTSNMP command on an MVS console or SDSF to start the SLA subagent as a started task. A sample procedure is shipped in member EZAPAGSN in SEZAINST.

The SLA Subagent can be stopped using the stop command (P PAGTSNMP) or using the kill command in the OS/390 shell. The following kill command with the TERM signal will enable the SLA subagent to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where *pid* is where pid is the pagtsnmo process ID.

The pagtsnmp process ID can be obtained using the following OS/390 UNIX command:

```
ps  -A
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the MIB objects supported by the SLA subagent.

Traffic Regulation Management

Traffic regulation management is a way to regulate the number of TCP/IP connections to any given port. This is described in the following sections.

Traffic Regulation (TR)

z/OS CS provides a traffic regulation manager for limiting TCP connections. This support can be used to take preemptive action to prevent overconsumption of S/390 resources if an unexpected increase in TCP connection requests occurs. Traffic Regulation can regulate the number of connections on a port basis based on the requester's IP address and current connection consumption, as well as the state of the system.

This is accomplished through TR policy, which is controlled by the Policy Agent (PAGENT). TCP traffic regulation policy requires the installation to specify two numbers for each TCP port to be protected:

- A hard limit of concurrent connections allowed for the port
- A percentage of the connections that can be used by a single host

The percentage is applied against the number of available connections for the port. Therefore, as fewer connections become available, each host is allowed fewer new connections. The percentage is applied against the number of available connections, rather than the total number of connections allowed, in order to allow access to a larger number of different hosts when resources are low.

When a host requests a connection, the number of connections it currently holds for the port is compared to the percentage applied to the connections currently available for the port. If the number currently held is less than the percentage of

currently available connections, the host is allowed to open an additional connection. If equal or greater, the host is not allowed to open further connections until more connections are freed up. All connection requesters for the port are regulated by this mechanism. The customer does not have to manually configure IP addresses.

For example, if there are currently four connections unused and the percentage specified in the policy is 50% then 4 (the number of available connections) times .5 equals two. This means at this point in time, a host that has two or more connections will not be able to obtain an additional connection. For example:

- If host A already has two connections, host A will not be able to get another one (50% of the unused connections are already held).
- However, if host B only has one connection, host B will be able to request another (because only 25% of the unused connections are currently held.)

If a host does not currently have any connections open on the port and unused connections are available, a host will always be allowed at least 1 connection.

There is one case where TR will allow a host more connections than defined by the TR policy. This can occur when QoS differentiated services policy is defined for the specific outbound client IP address (DestinationAddressRange) in question and the differentiated services policy would allow this IP address a higher number of connections (MaxConnections) than allowed by the TR policy. TR will honor the QoS differentiated services policy if TR is not in a *constrained state*. A connection that is allowed due to a higher QoS differentiated services limit is called a QoS exception by TR and if logging is requested, it will be logged as a TR event. TR is considered in a *constrained state* when 90% of the total number of connections (TotalConnections) are already held. After a *constrained state* is entered, TR remains in a *constrained state* mode until 12% of the total number of connections are again available. A QoS exception is made only when QoS differentiated service policy is applied for the specific source server port and specific outbound client destination IP address - if either of these attributes specify a range or are null, the QoS exception will not be made.

The following is a policy example where a QoS exception could occur. The TR policy alone will allow a maximum of nine [2] concurrent connections to any host that connects into port 23. However, because there is a QoS differentiated services policy for 9.27.85.9 that allows up to 15 connections for this IP address, 9.27.85.9 could obtain up to 15 connections if port 23 is not constrained.

```
PolicyRule port23
  {
   DestinationPortRange 23
   PolicyActionReference  ActionTN
   }
PolicyAction ActionTN
  {
  PolicyScope  TR
  TypeActions      Log  Limit
  TotalConnections 1000
  Percentage           1
  }
```

2. To understand why the maximum number is 9 (rather than 10) in the above policy example, let's look at the scenario where the only connections come from a single IP address. The first 9 connections work fine. When the 10th connection is attempted, the number of unused connections is 991. Apply the Percentage (1%) against the number of unused connections (991): .01 times 991 equals 9.91. The decimal places are dropped due to implementation, giving the maximum of 9 connections per host (based on TR policy).

```
PolicyRule qos1rule
  {
  SourcePortRange 23
  DestinationAddressRange 9.27.85.9
  PolicyActionReference qos1action
  }
PolicyAction qos1action
  {
  MaxConnections 15
  }
```

Similarly, if there is QoS differentiated services policy for an IP address that is more restrictive (that is, allows fewer connections for the IP address than the TR policy indicates), the differentiated services policy is honored. In this case, because differentiated services checks are implemented prior to the TR checks, TR may not see the connection request and TR statistics and event logs will not reflect a connection attempt that was rejected due to QoS differentiated services limits.

The percentage limit has to be tuned to the type of traffic the installation expects for the application. For example, if the installation is running an FTP server that is mainly for transfers between two hosts, the administrator would set the percentage high, so that host could use as many connections as is needed. Or, for the case of an internet web server, where there are many different hosts connecting, the percentage could be set very low, to allow many different hosts access as well as limit any flooding attempts from a single source to a few connections.

The traffic regulation policy can run in several different modes, which are defined as part of the TR action in the policy. These modes apply on a per-port basis.

Statistics
: Lets the installation gather baseline statistics or statistics based on an experimental policy. Statistics are reported on a policy-defined interval to syslogd.

Log
: Logs events that exceed the specified policy to syslogd. Constrained state and QoS limit exceptions are also logged.

Limit
: Enforces the policy by refusing connections that are outside of the policy limits.

To gather baseline statistics, an installation will first need to run in Statistics mode, with the traffic regulation daemon (trmd) running. In statistics mode, the following information is provided for the port on a policy defined interval:

- Total number of connections requested during the interval
- Total number of connections closed during the interval
- The IP address of the host that requested a connection during the interval and held the highest number of concurrent connections during the interval, and the highest number of concurrent connections held by this IP address
- A suggested value for TotalConnections based on this interval
- A suggested value for Percentage based on this interval

While the baseline statistics records provide suggested policy values for the interval, the installation should evaluate data from multiple intervals.

After the installation determines the policy values to use, try running with the Log action specified. Specifying the Log action, without the Limit action, basically tests out the policy. The connections that would have been denied (if the Limit action was

specified) are logged, but the connection is allowed to occur. After the installation is satisfied with the experimental policy, the policy action can be set to Limit.

See “Defining TRM Policies Using PAGENT” on page 420 for sample policy for these three stages.

The Traffic Regulation support contains three elements:

- A Pagent component that defines and activates the Traffic Regulation policy in the stack
- The Traffic Regulation daemon (TRMD), which obtains statistics and event information from the stack and logs information to syslogd (based on Traffic Regulation policy defined in Pagent)
- A utility (trmdstat), which reads the log file created by TRMD and provides summary reports for events that were logged

The actual Traffic Regulation process starts when the Pagent policy is activated, even if TRMD is not active. However, to obtain the statistics and event logging information, TRMD must be started. To start TRMD, the TCP stack and pagent must be active.

Traffic Regulation Actions

As mentioned above, the traffic regulation policy can run in the following modes that are defined as part of the TR action in the policy:

- Statistics
- Log
- Limit

These modes apply on a per-port basis.

Limit and Log modes require:

- The size of the total connection pool (TotalConnections)
- The percentage of available connections that a single host can hold (Percentage)

Only when *Limit* mode is specified will connections actually be refused. Log mode allows the user to try out a pool size and a percentage before actually applying them. Statistics mode allows the user to collect data without having to specify a pool size and percentage.

Log mode records each connection that exceeds the policy limit to syslogd. If the installation prefers to have these records written to the console, the syslogd configuration file should be modified to indicate /dev/console as the syslogd target.

These modes, combined with the connection pool size and the percentage, define the TR action. Modes can operate simultaneously. For example, Limit and Log can both be on at the same time. The following chart shows the result when modes are used simultaneously.

Mode Setting	Result
Log and Limit	Connections will be refused and those refusals will be logged.
Log and Statistics	Events will be logged according to the policy, and statistics will be gathered.

Mode Setting	Result
Limit and Statistics	Connections will be refused according to policy, and statistics will be gathered.
Log, Limit, and Statistics	Connections will be refused according to policy, those refusals will be logged, and statistics will be gathered.

Messages EZZ8484I through EZZ8493I are used for statistics and event log information. Refer to *z/OS Communications Server: IP Messages Volume 3 (EZY)* for a description of the information that is recorded for Statistics and Log data.

Configuring the TRMD

To enable this function, follow these steps:

1. Configure TR policy in the Policy Agent (PAGENT) configuration file.
2. Start PAGENT.
3. Start trmd.

Note: If log/statistics/debug data is required, ensure that syslogd is started before starting TRMD

Starting and Stopping the TRMD

TRMD runs as an authorized program and requires some RACF setup (TRMD must be able to run as a started task and have superuser authority). See the EZARACF member of hlq.SEZAINST for sample RACF commands.

The Policy Agent API libraries (papi.dll) must be accessible to TRMD. The papi.dll resides in usr/lpp/tcpip/lib. The LIBPATH environment variable can be set to indicate where papi.dll is found.

The TCP/IP stack and PAGENT must be running before TRMD can be started.

As described below, TRMD can be started from the OS/390 shell or as a started task.

Running TRMD as a Started Task

A sample procedure is shipped in member EZATRMD in SEZAINST. Follow the instructions in the sample member to define your environment.

To start TRMD as a started task, use the *S TRMD* command from the MVS console or SDSF. In some cases, TRMD issues a fork, and the job name will be the original job name with a number appended. For example, *S TRMD* might result in the TRMD started task running under the job name TRMD1. Use the *D A,TRMD** command to verify the job name that TRMD is running under.

If running as a started task, issue *P jobname* to stop TRMD.

Running TRMD from the OE Shell

Only a superuser can run TRMD from the OE shell. To run TRMD from the OE shell, set the LIBPATH environment variable to indicate where papi.dll is found. For example:

```
export LIBPATH=$LIBPATH:/usr/lpp/tcpip/lib
```

After the proper environment is set up, issue the following to start TRMD:

```
trmd
```

To stop TRMD, issue the following kill command :

```
kill -s TERM pid#
```

where pid is the TRMD process ID

To obtain the TRMD process ID, issue the following z/OS UNIX command:

```
ps  -A
```

Debug options can also be specified when starting TRMD. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

TRMDSTAT

Trmdstat is a utility program that runs from the OE shell. Trmdstat reads a log file, analyzes the log records generated by TRMD, and provides summary or detailed reports based on the options specified.

The following reports can be requested:

- Overall summary of TRMD logged events

  This report displays the summary count of out of policy conditions, QoS exception conditions, and constrained state conditions logged by TRMD.
- Summary of out of policy conditions by port
- Summary of out of policy conditions by port and source IP address
- Details of out of policy conditions for a requested source IP address

Refer to *z/OS Communications Server: IP User's Guide* for the complete the TRMDSTAT command and samples of the reports generated by TRMDSTAT.

Verification

To verify that policies are correctly defined and functioning properly, consider the following points:

- Are the policies defined correctly?
- Are the policies installed in the TCP/IP stack(s)?
- Is the expected traffic mapping to the correct policies?
- Are the Sysplex Distributor policy functions working correctly?
- Are the TRM policy functions working correctly?
- Does anything need to be tuned?

The following sections provide more details about these considerations.

Are the Policies Defined Correctly?

When starting the Policy Agent, first check for any error messages issued to the console. Message EZZ8434I indicates something is wrong with the Policy Agent environment. Message EZZ8438I indicates a syntax or semantic error in the policy definitions. Messages EZZ8439I and EZZ8440I indicate a problem with the LDAP server configuration or the server itself. Refer to *z/OS Communications Server: IP Diagnosis* for more information on these types of problems. Use the UNIX pasearch command to display policy definitions. The output from this command indicates whether or not policy rules are active, and shows the parsed results of the policy definition attributes. One thing to note is that the Policy Agent is designed to ignore unknown attributes, so misspelled attributes will result in default values being used. The pasearch output can be used to verify that policies are correctly defined.

Are the Policies Installed in the TCP/IP Stack(s)?

Use the NETSTAT SLAP or onetstat -j command to display policy statistics. This command only displays statistics for active (installed) policies, so can be used to verify the correct policies are installed, even if the statistics are all 0. Since the Policy Agent can install policies on multiple stacks, issue this command on each stack to verify the correct set of policies is installed.

For more information, see “Using NETSTAT to Display Active Policy Statistics” on page 451.

Is the Expected Traffic Mapping to the Correct Policies?

While connections are active, use the NETSTAT ALL or onetstat -A command to display details about the active connections. One piece of information displayed is the policy rule name. If this name is blank, then the traffic is not mapped to any active rule. Also, use the NETSTAT SLAP or onetstat -j command to display policy statistics. The output shows the time that each policy was last mapped to traffic, and accumulated statistics for each policy. Monitor these values over time to verify that new traffic is mapping as expected.

Note: The values displayed by the NETSTAT SLAP and onetstat -j commands can wrap around to 0. If some of the values do not seem correct (for example, total out bytes less than total out bytes in profile), then wrapping has probably occurred.

Are the Sysplex Distributor Policy Functions Working Correctly?

For Sysplex Distributor, you can enter NETSTAT VDPT on the distributing stack to display the WLM and WLM/QOS combined weight information for each target TCP/IP.

Refer to *z/OS Communications Server: IP Diagnosis* for more information.

Are the TRM Policy Functions Working Correctly?

To determine whether TR policy is applied correctly, the policy should include TypeActions Log and Statistics, and also start TRMD. The statistics and the log data then can be analyzed to determine whether the TR policy is applied correctly. The trmdstat command can be used to help analyze TR statistics and log data.

Does Anything Need to be Tuned?

When poor performance (for example, low throughput, long response time, and so on) is experienced unexpectedly and rather consistently by certain set of users/applications (or TRAPs are generated by the SLA MIB subagent), the problem might be in the way the QoS policy is defined for the corresponding set of users/applications. For example, the TOS/DS value might be set incorrectly to lower QoS level than it is intended (for example, medium/low priority instead of high). It is important to remember that given a fixed amount of network resources, changing some traffic demand from low to higher QoS level will mean that other traffic demands will be effected; therefore, use care to ensure that in attempting to meet one set of QoS requirement, other (or worse) problems do not result.

Another cause for poor performance might be in the way the bandwidth allocation defined via the DiffServ parameters or TCP maxrate/minrate is not adequate to accommodate the traffic demand. Yet, another possibility might be that either network or the server capacity is not adequate to handle the traffic demand. This is

evident when a majority of users/applications do not have their QoS requirements met. When this happens, network planning process must be revisited.

For more information, see “Using the SLA Subagent to Monitor Policies” on page 445.

Using PASEARCH to Display All Active and Inactive Policies

The following command displays all active and inactive policies:

```
pasearch -AI
```

```
MVS TCP/IP pasearch CS V2R10             TCP/IP Image:      TCPCS
  Date:                 03/23/2000       Time:  06:37:34

policyRule:             DiffServ_Rule1
  Version:              2                Status:            Inactive
  Distinguish Name:     pr=DiffServ_Rule1,g=policy,o=IBM,c=US
  Priority:             0                Sequence Actions:  Don't Care
  ConditionListType:    DNF              No. Policy Action: 1
  policyAction:         DiffServ_Action1
   ActionType:          QOS              Action Sequence:   0
  Time Periods:
   Day of Month Mask:   11111111110000000000000000000000
   Month of Year Mask:  111111111111
   Day of Week Mask:    0111110  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   From TimeOfDay:      06:00            To TimeOfDay:      20:00
   From TimeOfDay UTC:  11:00            To TimeOfDay UTC:  01:00
   TimeZone:            Local
  Condition Summary:                    Negative Indicator:  OFF
   RouteCondition:
    InInterface:        0.0.0.0          OutInterface:      9.67.116.98
   HostCondition:
    SourceIpFrom:       0.0.0.0          SourceIpTo:        0.0.0.0
    DestIpFrom:         211.40.100.0     DestIpTo:          211.40.100.255
   ApplicationCondition:
    ProtocolNumFrom:    0                ProtocolNumTo:     0
    SourcePortFrom:     20               SourcePortTo:      21
    DestPortFrom:       5000             DestPortTo:        9999
    ApplicationName:    FTP*
    ApplicationData:

  Qos Action:           DiffServ_Action1
    Version:            2                Status:            Active
    Distinguish Name:   pa=DiffServ_Action1,g=policy,o=ibm,c=US
    Scope:              DataTraffic      OutgoingTOS:       01000000
    Permission:         Allowed
    MaxRate:            0                MinRate:           0
    MaxDelay:           50               MaxConn:           100
    Routing Interfaces: 0
    RSVP Attributes
     ServiceType:       0                MaxRatePerFlow:    0
     MaxTokBuckPerFlw:  0                MaxFlows:          0
    DiffServ Attributes
     InProfRate:        128              InProfPeakRate:    0
     InProfTokBuck:     32               InProfMaxPackSz:   0
     OutProfXmtTOSByte: 00000000         ExcessTrafficTr:   BestEffort
    TR Attributes
     TotalConnections:  0                LoggingLevel:      0
     Percentage:        0                TimeInterval:      0
     TypeActions:       0
```

Using PASEARCH to Display All Policies With a Given Name

The following command displays all policies whose names begin with ″ftp″:

```
pasearch

MVS TCP/IP pasearch CS V2R10             TCP/IP Image:      TCPCS
  Date:                 03/23/2000       Time:  06:53:32

policyRule:             RSVP_Rule1
  Version:              2                Status:            Active
  Distinguish Name:     pr=RSVP_Rule1,g=policy,o=IBM,c=US
  Priority:             0                Sequence Actions:  Don't Care
  ConditionListType:    DNF              No. Policy Action: 1
  policyAction:         RSVP_Action1
   ActionType:          QOS              Action Sequence:   0
  Time Periods:
   Day of Month Mask:   11111111111111111111111111111111
   Month of Year Mask:  111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   From TimeOfDay:      00:00            To TimeOfDay:      24:00
   From TimeOfDay UTC:  05:00            To TimeOfDay UTC:  05:00
   TimeZone:            Local
  Condition Summary:                    Negative Indicator:  OFF
   RouteCondition:
    InInterface:        0.0.0.0          OutInterface:      0.0.0.0
   HostCondition:
    SourceIpFrom:       0.0.0.0          SourceIpTo:        0.0.0.0
    DestIpFrom:         0.0.0.0          DestIpTo:          0.0.0.0
   ApplicationCondition:
    ProtocolNumFrom:    6                ProtocolNumTo:     6
    SourcePortFrom:     8000             SourcePortTo:      8001
    DestPortFrom:       0                DestPortTo:        0
    ApplicationName:
    ApplicationData:

  Qos Action:           RSVP_Action1
    Version:            2                Status:            Active
    Distinguish Name:   pa=RSVP_Action1,g=policy,o=ibm,c=US
    Scope:              RSVP             OutgoingTOS:       01100000
    Permission:         Allowed
    MaxRate:            0                MinRate:           0
    MaxDelay:           0                MaxConn:           0
    Routing Interfaces: 0
    RSVP Attributes
     ServiceType:       ControlledLoad   MaxRatePerFlow:    400
     MaxTokBuckPerFlw:  48               MaxFlows:          10
    DiffServ Attributes
     InProfRate:        0                InProfPeakRate:    0
     InProfTokBuck:     0                InProfMaxPackSz:   0
     OutProfXmtTOSByte: 00000000         ExcessTrafficTr:   BestEffort
    TR Attributes
     TotalConnections:  0                LoggingLevel:      0
     Percentage:        0                TimeInterval:      0
     TypeActions:       0
```

Using PASEARCH to Display All Policies with FTP

The following command displays all policies whose name starts with FTP:

```
pasearch -wf ftp

MVS TCP/IP pasearch CS V2R10             TCP/IP Image:      TCPCS
  Date:                 03/23/2000       Time:  06:59:31
```

```
policyRule:             ftprule
  Version:              2                 Status:            Active
  Distinguish Name:     pr=ftprule,pg=groupA-1,g=policy,o=ibm,c=US
  Group Distinguish Nm: pg=groupA-1,g=policy,o=ibm,c=US
  Priority:             2                 Sequence Actions:  Mandatory
  ConditionListType:    DNF               No. Policy Action: 1
  policyAction:         ftpaction
   ActionType:          QOS               Action Sequence:   1
  Time Periods:
   Day of Month Mask:   11111111111111111111111111111111
   Month of Year Mask:  111100111100
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time UTC: Tue Jul 13 00:00:00 1999
   End Date Time UTC:   Wed Oct 30 20:00:00 2002
   From TimeOfDay UTC:  02:00             To TimeOfDay UTC:  20:00
   TimeZone:            UTC
  Condition Summary:                      Negative Indicator:  OFF
   RouteCondition:
    InInterface:        0.0.0.0           OutInterface:      0.0.0.0
   HostCondition:
    SourceIpFrom:       9.37.80.0         SourceIpTo:        9.37.80.255
    DestIpFrom:         0.0.0.0           DestIpTo:          0.0.0.0
   ApplicationCondition:
    ProtocolNumFrom:    6                 ProtocolNumTo:     6
    SourcePortFrom:     0                 SourcePortTo:      0
    DestPortFrom:       20                DestPortTo:        21
    ApplicationName:
    ApplicationData:

  Qos Action:           ftpaction
    Version:            2                 Status:            Active
    Distinguish Name:   pa=ftpaction,pg=groupA-1,g=policy,o=ibm,c=US
    Scope:              DataTraffic       OutgoingTOS:       01000000
    Permission:         Allowed
    MaxRate:            400               MinRate:           0
    MaxDelay:           0                 MaxConn:           50
    Routing Interfaces: 3
      Interface Number: 1                 Interface:         9.67.116.1
      Interface Number: 2                 Interface:         9.67.116.2
      Interface Number: 3                 Interface:         0.0.0.0
    RSVP Attributes
     ServiceType:       0                 MaxRatePerFlow:    0
     MaxTokBuckPerFlw:  0                 MaxFlows:          0
    DiffServ Attributes
     InProfRate:        0                 InProfPeakRate:    0
     InProfTokBuck:     0                 InProfMaxPackSz:   0
     OutProfXmtTOSByte: 00000000          ExcessTrafficTr:   BestEffort
    TR Attributes
     TotalConnections:  0                 LoggingLevel:      0
     Percentage:        0                 TimeInterval:      0
     TypeActions:       0
```

Using the SLA Subagent to Monitor Policies

SLA Subagent Performance Monitoring

The SLA Subagent provides information about service policies and performance data for applications mapped to those policies via two tables.

Note: The SLA subagent can be used to monitor Differentiated Services policies, and RSVP reservations if a corresponding RSVP policy is defined.

slapmPolicyRuleStatsTable

Provides information about defined service policies and aggregate performance data for mapped applications.

slapmSubcomponentTable
Provides information about individual TCP or UDP applications and application-specific performance data.

The SLA Subagent also supports performance monitoring via the slapmPRMonTable object. Entries are created in the monitor table to establish the desired criteria for monitoring. Two levels of monitoring are provided:

Aggregate
Monitoring is performed based on the aggregate of all TCP or UDP applications that are mapped to one or more service policies.

Subcomponent
Monitoring is performed based on a single TCP or UDP application.

Three types of monitoring are provided for measuring application performance:

MinRate
The current input/output rates of the application(s) are compared to threshold values established in the monitor table entry. If the current rates are less than the threshold, an SNMP trap is sent if traps are enabled.

MaxRate
The current input/output rates of the application(s) are compared to threshold values established in the monitor table entry. If the current rates are greater than the threshold, an SNMP trap is sent if traps are enabled.

MaxDelay
The current delay rates of the application(s) are calculated by using TCP round trip time (RTT). For aggregate monitoring, the RTT of all TCP applications are averaged. The delay rates are compared to threshold values established in the monitor table entry. If the current rates are greater than the threshold, an SNMP trap is sent if traps are enabled.

Note: MaxDelay monitoring is only available for TCP applications.

Refer to RFC 2758 for the SLAPM MIB in the sample file slapm.text in the /usr/lpp/tcpip/samples directory for more details about how to make the various monitoring calculations.

When SNMP traps are enabled, and a *not achieved* trap is sent as described above, a corresponding *okay* trap is sent when the traffic once again conforms to the boundaries established in the monitor table entry.

For example, you can establish MaxDelay monitoring and use a max delay low value of 50 and a max delay high value of 75. If the RTT of the application(s) rises above 75, a *not achieved* trap is sent. If the RTT then drops below 50, an *okay* trap is sent to indicate the problem has been resolved. However, if the application(s) end before conforming to the established boundaries, an okay trap may not be sent naturally. For application level traps, the okay trap is never sent because the corresponding subcomponent table entry is deleted when the application(s) end, which also removes the application level monitoring.

For aggregate traps, when the application(s) end, MaxRate and MaxDelay okay traps will be sent, because a value of 0 for each of these should fall below the minimum values established in the monitor table entry. Conversely, for MinRate monitoring, an aggregate okay trap is not sent, because a value of 0 will never be greater than the maximum value established in the monitor table.

In addition to the traps used to measure application performance, two additional traps are used to monitor table administration:

Policy Deleted
Trap is sent when an entry is deleted from the slapmPolicyRuleStatsTable.

Monitor Deleted
Trap is sent when an entry is deleted from the slapPRMonTable.

Creating Monitor Table Entries and Enabling SNMP Traps: Several MIB objects are used when establishing monitor table entries and for configuring whether and how often traps are sent. First, to establish monitor table entries, set the following MIB object variables.

Note: Because most of these objects have default values, you might be able to achieve the desired monitoring using only a subset of the objects.

slapmPRMonControl Controls what levels and types of monitoring are in effect.

slapmPRMonInterval Sets the interval for calculating input/output and delay rates and checks those values against the monitor table thresholds.

slapmPRMonMinRateLow, slapmPRMonMinRateHigh, slapmPRMonMaxRateLow, slapmPRMonMaxRateHigh, slapmPRMonMaxDelayLow, slapmPRMonMaxDelayHigh
Establishes the threshold values for MinRate, MaxRate, and MaxDelay monitoring. The min/max rates are in units of kilobits per second, and the max delay is in units of milliseconds.

slapmPRMonRowStatus Controls the status of a monitor table entry, for instance whether or not the entry is active.

In addition, the following MIB objects are used to control the generation of traps:

slapmPolicyTrapEnable
Enables or suppresses generation of Policy Deleted and Monitor Deleted traps.

slapmPolicyTrapFilter
Establishes the number of times a given MinRate, MaxRate, or MaxDelay event must be encountered before a trap is generated.

slapmPolicyPurgeTime
Establishes a timeout value for Policy Deleted traps. After a service policy is deleted, the amount of time indicated by this object must expire before a Policy Deleted trap is generated.

slapmPRMonControl
Controls whether or not aggregate and/or subcomponent traps are enabled.

Creating the Monitor Table Index: When you create monitor table entries, specify the appropriate index value. The index is composed of the following

- slapmPRMonOwnerIndex
- slapmPRMonSystemAddress
- slapmPRMonIndex

The OwnerIndex is expressed in the following format:

```
length.character.character...
```

where character is in ASCII decimal form.

For example, the value ″u1″ is expressed as ″2.117.31″. The SystemAddress value will always be 0. The Index value is the index into the slapmPolicyNameTable that maps to the policy name.

For example, assume the following service policy is defined:

```
PolicyAction                ElaineCat
     {
          PolicyScope             RSVP
          MaxRatePerFlow          640
          MaxTokenBucketPerFlow   440
          Maxdelay                1
     }
     PolicyRule                   ElainePol
     {
          ProtocolNumberRange     6
          SourcePortRange         8000
          PolicyActionReference   ElaineCat
     }
```

The SLA Subagent will create an entry in the slapmPolicyNameTable to represent the policy rule. The index value for this entry is arbitrary and assigned by the subagent. Corresponding entries in the other MIB tables, including the monitor table, contain the index value that maps to the entry in the name table.

To assist you in creating the index for the monitor table entries, note that the index value used in the slapmPolicyRuleStatsTable entries consists of the last two values used in the monitor table index, namely the SystemAddress and Index. Thus, you can walk through the policy statistics table using the following command:

```
osnmp -v walk slapmPolicyRuleStatsTable
```

Then, cut-and-paste the index value from the PolicyRuleStatsTable and add an OwnerIndex of your choosing at the beginning of the index.

For the above example, the complete index using an OwnerIndex of ″u1″ is:

```
2.117.31.0.3
|         | +--- name table index value (Index)
|         +----- no SystemAddress (SystemAddress)
+--------------- length + "u1" (OwnerIndex)
```

Monitor Table Examples:

Note: If you are going to change any of the monitor table object values for an existing table entry or row, you must take the row out of service to make the changes. To do this, set the value of slapmPRMonRowStatus to 2.

After your changes are made, set the row status to a value of 1 to put it back in service.

Two of the monitor table objects, monitor control and monitor status fields, are important in setting up the entries and understanding why traps are generated. Both of these fields have the SNMP data type BITS, which means they are bit strings, where bit 0 is the low order bit. Any combination of bits can be set into these objects.

Table 17 shows the meaning of the various bits:

Table 17. Monitor Control and Monitor Status Object Bit Values

slapmPRMonControl Object	**xx54 3210**
0 - monitor MinRate	0000 0001 = 01
1 - monitor MaxRate	0000 0010 = 02
2 - monitor MaxDelay	0000 0100 = 04
3 - enable aggregate traps	0000 1000 = 08
4 - enable subcomponent traps	0001 0000 = 10
5 - monitor subcomponent	0010 0000 = 20
slapmPRMonStatus Object	xx98 7654 3210
0 - slaMinInRateNotAchieved	0000 0000 0001 = 001
1 - slaMaxInRateExceeded	0000 0000 0010 = 002
2 - slaMaxDelayExceeded	0000 0000 0100 = 004
3 - slaMinOutRateNotAchieved	0000 0000 1000 = 008
4 - slaMaxOutRateExceeded	0000 0001 0000 = 010
5 - monitorMinInRateNotAchieved	0000 0010 0000 = 020
6 - monitorMaxInRateExceeded	0000 0100 0000 = 040
7 - monitorMaxDelayExceeded	0000 1000 0000 = 080
8 - monitorMinOutRateNotAchieved	0001 0000 0000 = 100
9 - monitorMaxOutRateExceeded	0010 0000 0000 = 200

The following examples show how to create monitor table entries to monitor at various levels and types. You can also create other combinations using the monitor control object.

This example assumes SNMP version 1 security and no SNMPD.CONF file.

First, enable traps, assuming Version 1 security and no SNMPD.CONF file. The snmptrap.dest file should contain the IP address and protocol of an entity to receive traps. In this example, use the osnmp command running in the background to receive traps.

```
/etc/snmptrap.dest   contains: 9.67.191.5    UDP
  /etc/pw.src          contains: public 0.0.0.0    0.0.0.0

  osnmp set slapmPolicyTrapEnable.0 1
  osnmp set slapmPolicyTrapFilter.0 1
  osnmp set slapmPolicyPurgeTime.0 60
  osnmp trap > /tmp/trap.output &
```

To monitor for MinRate at an aggregate level:

```
osnmp set slapmPRMonControl.index \'00000009\'h
           where  9 is aggregate monitor and trap for MinRate
                  1 is aggregate monitor for MinRate

  osnmp set slapmPRMonMinRateLow.index  l
  osnmp set slapmPRMonMinRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MinRate occurs, then look at the monitor status object in the trap, or:

```
  osnmp get slapmPRMonStatus.index
           should be 1 if inbound MinRate not achieved
                     8 if outbound MinRate not achieved
```

To monitor for MaxRate at an aggregate level:

```
osnmp set slapmPRMonControl.index \'0000000a\'h
           where  a is aggregate monitor and trap for MaxRate
                  2 is aggregate monitor for MaxRate

  osnmp set slapmPRMonMaxRateLow.index  l
  osnmp set slapmPRMonMaxRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 2 if inbound MaxRate not achieved
                    10 if outbound MaxRate not achieved
```

To monitor for MaxDelay at an aggregate level:

```
osnmp set slapmPRMonControl.index \'0000000c\'h
           where  c is aggregate monitor and trap for MaxDelay
                  4 is aggregate monitor for MaxDelay

  osnmp set slapmPRMonMaxDelayLow.index  l
  osnmp set slapmPRMonMaxDelayHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxDelay occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 4 if MaxDelay exceeded
```

To monitor for MinRate at an application level:

```
osnmp set slapmPRMonControl.index \'00000031\'h
           where 31 is subcomponent monitor and trap for MinRate
                 21 is subcomponent monitor for MinRate

  osnmp set slapmPRMonMinRateLow.index  l
  osnmp set slapmPRMonMinRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MinRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 20 if inbound MinRate not achieved
                    100 if outbound MinRate not achieved
```

To monitor for MaxRate at an application level:

```
osnmp set slapmPRMonControl.index \'00000032\'h
           where 32 is subcomponent monitor and trap for MaxRate
                 22 is subcomponent monitor for MaxRate

  osnmp set slapmPRMonMaxRateLow.index  l
```

```
  osnmp set slapmPRMonMaxRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 40 if inbound MinRate not achieved
                    200 if outbound MinRate not achieved
```

To monitor for MaxDelay at an application level:

```
osnmp set slapmPRMonControl.index \'00000034\'h
           where 34 is subcomponent monitor and trap for MaxDelay
                 24 is subcomponent monitor for MaxDelay

  osnmp set slapmPRMonMaxDelayLow.index  l
  osnmp set slapmPRMonMaxDelayHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxDelay occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 80 if MaxDelay exceeded
```

Using NETSTAT to Display Active Policy Statistics

The following command displays active policy statistics:

```
netstat -j

MVS TCP/IP onetstat CS V2R10       TCPIP Name: TCPCS            08:01:27
PolicyRuleName:  TCPBasic
  FirstActTime:    12:57:26      LastMapTime:      12:59:10
  TotalBytesIn:    0000033328    TotalBytesOut:    0000030844
  BytesInDiscard:  0000000000    BytesOutDiscard:  0000000000
  TotalInPackets:  0000000124    TotalOutPackets:  0000000130
  ActConnMap:      0000000000    MaxConnLimit:     0000000000
  AcceptConn:      0000000011    DeniedConn:       0000000000
  OutBytesInProf:  0000000000
PolicyRuleName:  UDPBasic
  FirstActTime:    12:57:26      LastMapTime:      13:00:50
  TotalBytesIn:    0000031052    TotalBytesOut:    0000030756
  BytesInDiscard:  0000000000    BytesOutDiscard:  0000000000
  TotalInPackets:  0000000037    TotalOutPackets:  0000000037
  ActConnMap:      0000000000    MaxConnLimit:     0000000000
  AcceptConn:      0000000000    DeniedConn:       0000000000
  OutBytesInProf:  0000000000
```

Chapter 10. Network Management

This chapter describes how to configure:

- Simple Network Management Protocol Agent (osnmpd)
- z/OS UNIX SNMP command (osnmp)
- NetView SNMP command
- Subagents
- Trap forwarder daemon

Before you configure, read “Understanding Search Orders of Configuration Information” on page 6. It covers important information about data set naming and search sequences. The z/OS UNIX osnmp command is the SNMP command used to access MIB object information from the z/OS shell, as the NetView SNMP command does from NetView.

For an overview of the functional components of SNMP, refer to the SNMP information in *z/OS Communications Server: IP Migration*.

Processing an SNMP Request

Figure 52 illustrates the interface between TCP/IP and the implementation of SNMP.

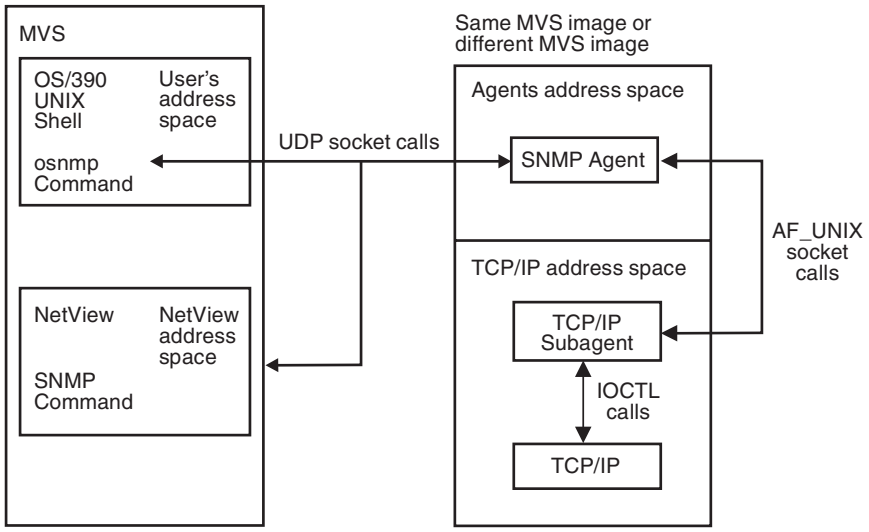


Figure 52. Overview of SNMP Support

This list illustrates the sequence of events from the time you issue an SNMP command until you receive the response:

1. The user issues a NetView SNMP (SNMP) or z/OS UNIX SNMP (osnmp) command.
2. The command processor validates and encodes the request in a Protocol Data Unit (PDU), and sends it to the SNMP agent.
3. The SNMP agent validates the request and, if necessary, sends it to an SNMP subagent. Requests for agent-oriented objects are handled by the agent and all others are handled by a subagent. To determine which objects are handled by the agent and which by a subagent, refer to the *z/OS Communications Server: IP User's Guide*.

4. The agent sends the response to the originator of the request. The command processor displays the response.

Note: Although not shown in Figure 52 on page 453, other subagents, such as the OMPROUTE subagent shipped as part of z/OS CS, also communicate with the SNMP agent using AF_UNIX socket calls or TCP socket calls from their own address spaces.

The SNMP agent and the SNMP subagents record trace information via the z/OS UNIX syslog daemon using the *daemon* facility. For detailed information regarding syslogd and specifying the daemon facility in the /etc/syslog.conf configuration file, see "Logging of System Messages" on page 25.

Deciding on SNMP Security Needs

The SNMP agent supports SNMPv1, SNMPv2c, and SNMPv3 security. SNMPv1 and SNMPv2c are community-based security, where a community name (or password) is passed with a request. If the community name is recognized as one that can be used by the IP address from which the request originates, the SNMP agent processes the request.

SNMPv3 provides a more powerful and flexible framework for message security and access control. Message security involves providing:

- Data integrity checking, to ensure that the data was not altered in transit
- Data origin verification, to ensure that the request or response originates from the source from which it claims to have come
- Message timeliness checking and, optionally, data confidentiality, to protect against eavesdropping

Access control is the ability to control exactly what data an individual user can read or write.

The SNMPv3 architecture introduces the User-Based Security Model (USM) for message security and the View-Based Access Control Model (VACM) for access control. The architecture supports the concurrent use of different security, access control, and message processing models. For example, community-based security can be used concurrently with USM.

USM uses the concept of a user for which security parameters (levels of security, authentication and privacy protocols, and keys) are configured at both the agent and the manager. Messages sent using USM are better protected than messages sent with community-based security, where passwords are sent in the clear and displayed in traces. With USM, messages exchanged between the manager and the agent have data integrity checking and data origin authentication. Message delays and message replays (beyond what happens normally due to a connection-less transport protocol) are protected against with the use of time indicators and request IDs. Data confidentiality, or encryption, is also available.

The use of VACM involves defining collections of data (called views), groups of users of the data, and access statements that define which views a particular group of users can use for reading, writing, or receipt in a notification.

SNMPv3 also introduces the ability to dynamically configure the SNMP agent using SNMP SET commands against the MIB objects that represent the agent's

configuration. This dynamic configuration support enables addition, deletion, and modification of configuration entries either locally or remotely. Remote modification of user keys can be especially useful.

Decide on your security needs—community-based or user-based.

If you are satisfied with the security of your existing configuration, you can continue to use community-based security with no migration. If you would like to take advantage of USM or VACM, you will need to migrate your configuration. Note that USM can be used only when both the SNMP agent and the manager requesting the data support USM, as the z/OS CS SNMP agent and the osnmp command do. VACM can be used even for community-based requests, but doing so requires migration of existing community name and trap destination definitions. Following is a list of the advantages and disadvantages of using each type of security.

Table 18. Security Advantages and Disadvantages

SNMPv1/SNMPv2c advantages	**SNMPv3 disadvantages**
Traditional standards-based administrative model	Emerging standards-based administrative model
Widely implemented on many platforms	Not yet implemented on many platforms
Easy to configure	More robust configuration options
SNMPv1/SNMPv2c disadvantages	**SNMPv3 advantages**
SNMPv1 and SNMPv2c allow particular IP addresses to access all data or no data	SNMPv3 allows a particular user to access particular data
Not very robust (password sent in PDU)	Robust (data integrity and data origin authentication)
Any user that can read data can also change the data (for objects defined as read-write).	The ability to change data can be limited to specific users
No data confidentiality	Encryption available
Configuration changes require restarting of SNMP agent	Configuration changes for USM and VACM can be made dynamically, either locally or remotely

For more information about security, see “Creating User Keys” on page 462.

Complete the following steps to configure SNMP:

1. Configure the SNMP agent (OSNMPD).
2. Configure the SNMP commands:
   - SNMP for NetView SNMP
   - osnmp for z/OS UNIX SNMP
3. Configure the SNMP subagents.
4. Configure the ATM Open Systems Adapter 2 (ATM OSA) support.
5. Configure the trap forwarder daemon.

Step 1: Configure the SNMP Agent (OSNMPD)

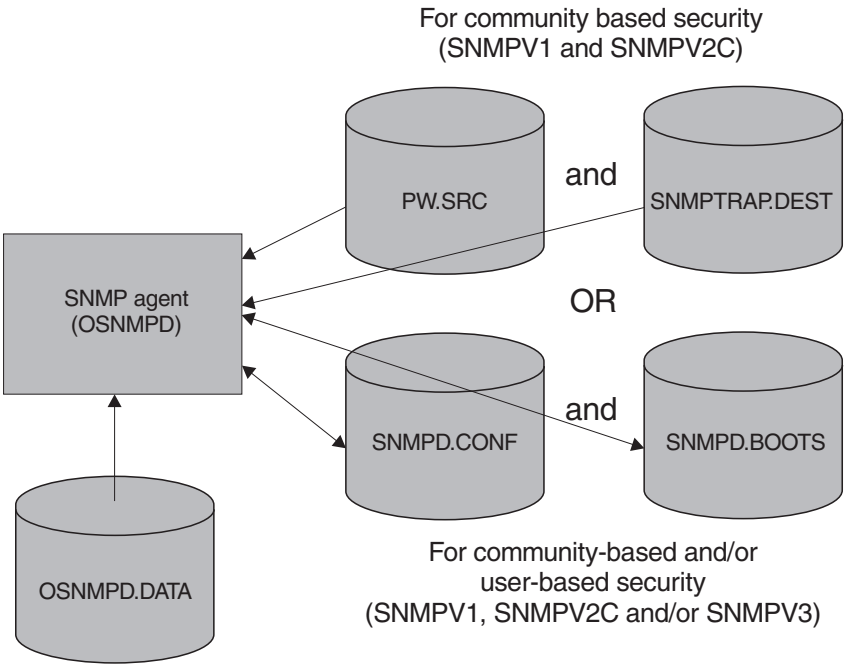


Figure 53. Configuration Files for SNMP Agent

Configure the SNMP agent based upon your security need. The SNMP agent accepts both SNMPv1 and SNMPv2c requests for community-based security. The SNMP agent can be configured to also use the User-based Security Model and the View-based Access Control Model. To configure the SNMP agent, perform the following tasks:

- "Provide TCP/IP Profile Statements"
- Depending upon whether you want to use USM and VACM, do one of the following:
  - If you are using community-based security and do not need USM or VACM, see "Provide Community-Based Security and Notification Destination Information" on page 458.
  - If you want the flexibility of using USM or VACM or community-based security, see "Provide Community-Based and User-Based Security and Notification Destination Information" on page 460.
- "Provide MIB Object Configuration Information" on page 463
- Refer to *z/OS Communications Server: IP Configuration Reference* for more information about OSNMPD parameters.

Provide TCP/IP Profile Statements

Update the following configuration statements in *hlq*.PROFILE.TCPIP:

AUTOLOG
PORT

There are two primary TCP/IP ports used by the SNMP agent, one for receiving incoming requests and one for sending traps to managers.

The default port used by the SNMP agent to receive incoming requests is 161. If you want the agent to use port 161 for this purpose and want to insure that no other application uses this port, you must specify the following PORT statement in your profile data set:

```
PORT
  161 UDP OSNMPD              ; SNMP Agent port for SNMP requests
```

If the agent will be started from the z/OS shell, reserve the port instead for z/OS UNIX by typing *OMVS* instead of *OSNMPD*.

If you want to define a port other than 161 for SNMP requests, you must do the following:

1. Start the agent with a `-p` parameter.
2. Configure management applications to use the new port:
   - For the osnmp command, make an entry in the OSNMP.CONF file with the correct port number. For details on creating this entry, see the description for *targetAgent* in the OSNMP.CONF statement in the *z/OS Communications Server: IP Configuration Reference*.
   - Where supported, configure other management applications to use the new port.
3. Configure subagents to use the new port:
   a. Specify the port number to use on the SACONFIG profile statement for the TCP/IP subagent.
   b. Specify the port number to use on the ROUTESA_CONFIG profile statement for the OMPROUTE subagent.
   c. Specify the port number to use on the -p parameter when starting the SLA subagent.
   d. If you are using DPI subagents other than those supplied with z/OS CS, set the SNMP_PORT environment variable to enable user-written subagents to connect to the agent.

The SNMP agent uses port 162, by default, for sending traps to the managers specified in SNMPTRAP.DEST or SNMPD.CONF file. Port 162 should be reserved for the management application primarily responsible for trap processing. If your environment requires multiple management applications at the same IP address to receive traps, consider using the Trap Forwarder Daemon. See “Step 5: Configure the Trap Forwarder Daemon” on page 474 for more details. If the SNMP query engine is typically used for processing traps and other applications, such as osnmp, are only occasionally used, the following port reservations are recommended.

```
PORT
  162 UDP SNMPQE  ; SNMPQuery Engine
```

You must also reserve additional ports for use by the osnmp command by specifying

```
nnnnn UDP OMVS
```

where *nnnnn* is a number in the range 0–65535 and *nnnnn* is used as the -p parameter value on the osnmp trap command.

If you want the SNMPQE and OSNMPD address spaces to be started automatically when the TCPIP address space is started, then include SNMPQE and OSNMPD in the AUTOLOG statement:

```
AUTOLOG
  SNMPQE                    ; SNMP Query Engine
  OSNMPD                    ; SNMP Agent
ENDAUTOLOG
```

Provide Community-Based Security and Notification Destination Information

If you are using only community-based security without the view-based access control model, do the following to configure the security and trap destinations.

Provide Community Name Information

SNMP agents are accessed by remote network management stations. To allow network management stations to send inquiries to the SNMP agent, you may provide PW.SRC information that defines a list of community names and IP addresses that can use these community names. The community name operates as a password when accessing objects on a destination SNMP agent.

The PW.SRC information is optional. If no PW.SRC information is found and no community name is specified for the -c parameter at agent invocation, then the SNMP agent will accept requests with a community name of 'public' from any IP address. If a PW.SRC file exists, but is empty, and if no community name is specified on the `-c` parameter at the agent invocation, then no requests will be accepted by the agent.

Note: Verify that there is no SNMPD.CONF file because this file can only be used with SNMPv3. If an SNMPD.CONF file is found, the PW.SRC file will not be used.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

PW.SRC Example: The PW.SRC statements could be specified as follows:

```
passwd1 9.0.0.0      255.0.0.0
passwd2 129.34.81.22 255.255.255.255
```

The PW.SRC statements specify community names and hosts that can use each community name. The format of a statement is:

community_name desired_network snmp_mask

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

The community name of an incoming SNMP request is compared to the known community names. If a match is found, then the IP address of the incoming request is logically ANDed with the *snmp_mask* of the PW.SRC statement. The result of the logical ANDing process is compared with the *desired_network*. If they match, the request is accepted.

In the preceding example, if a request for *community_name* `passwd1` is received from the IP address 9.34.22.122, IP address 9.34.22.122 is ANDed with 255.0.0.0. The result is 9.0.0.0, which equals the specified *desired_network* for `passwd1`, so this request is accepted. In `passwd2`, if the *community_names* match, only requests from host 129.34.81.22 are accepted.

If the *community_name* values do not match, or the IP address ANDed with the *snmp_mask* does not match, an `AUTHENTICATION_FAILURE` trap is sent if both of the following are true:

- A destination entry exists in SNMPTRAP.DEST.
- Authentication failure traps have been enabled. These traps are enabled by setting MIB object ″snmpEnableAuthenTraps.0″ to 1.

A *desired_network* and *snmp_mask* of all zeros allows anyone with the correct *community_name* to make requests.

Note: By default, the SNMP agent and the osnmp command send packets such that a VIPA address will be used as the originating address in the packet, if SOURCEVIPA is configured. This is a change introduced in V2R10; previously, the SNMP agent and the osnmp command set a socket option to cause the physical interface addresses to be used as the originating addresses on packets they sent. That meant the PW.SRC file had to contain all of the possible physical interface addresses that might be used, rather than a smaller number of VIPA addresses. A customer can override this change in behavior, if desired, by invoking the SNMP agent with the -a option or by using either the -a option or the NOSVIPA option in the osnmp command's OSNMP.CONF configuration file.

Provide Trap Destination Information

Traps are unsolicited messages that are sent by an SNMP agent to an SNMP network management station. An SNMP trap contains information about a significant network event. The management application running at the management station interprets the trap information sent by the SNMP agent.

The following traps are agent-generated:

- Authentication failure
- Cold start

The following traps are generated by the TCP/IP subagent:

- Link down
- Link up
- PVC creation
- PVC deletion

  PVC traps are reported for ATM Permanent Virtual Connections. For further information, see "Step 4: Configure the Open Systems Adapter (OSA) Support" on page 470.

The following traps are generated by the SLA subagent:

- Monitored event not achieved for aggregate policy
- Monitored event okay for aggregate policy
- Monitored even not achieved for individual connection policy
- Monitored event okay for individual connection policy
- Policy deleted
- Monitor table entry deleted

Note: The SNMP agent Distributed Protocol Interface allows subagents other than those shipped with z/OS CS (which might be running on another host) to

generate SNMP traps. This can allow for support of other types of traps. For more information about SNMP DPI, see the *z/OS Communications Server: IP Programmer's Reference*.

For additional detail on these traps, refer to the SNMP Trap Types information in *z/OS Communications Server: IP User's Guide*.

To use traps, you must provide SNMPTRAP.DEST information defining a list of managers to which traps are sent. The SNMPTRAP.DEST information is optional. If no trap destination file is found, then the SNMP agent sends traps to the IP address of the SNMP agent and issues a warning message indicating that defaults are in effect. If a trap destination file exists, but is empty, no traps are sent.

Note: Verify that there is no SNMPD.CONF file. If an SNMPD.CONF file is found, the SNMPTRAP.DEST file will not be used.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

SNMPTRAP.DEST Example: The SNMPTRAP.DEST statements could be specified as follows:

```
# SNMP Trap Destination information
124.34.216.1 UDP
MVSSYS2      UDP
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Provide Community-Based and User-Based Security and Notification Destination Information

If you want to use user-based security, either concurrently with or instead of community-based security, you must configure security definitions and notification destinations.

SNMPv3 provides the ability to configure the agent dynamically, from either a local or remote host, and to make changes in the configuration while the SNMP agent is running. Doing SNMP agent configuration dynamically requires a good understanding of how the SNMP SET commands can be issued to create new rows or to change or delete existing rows, as well as familiarity with the SNMP engine configuration tables defined in RFCs 2571 through 2575. (For additional detail, see RFCs 2571 through 2575, as well as RFCs 1901 through 1910.)

As an alternative to dynamically configuring the SNMP agent, z/OS CS supports a configuration file to be read at agent initialization called the SNMPD.CONF file. Dynamic configuration changes made with SNMP SET commands to the SNMP agent configuration entries will be written out to the SNMPD.CONF file, so they will continue to be in effect even after the SNMP agent is restarted.

SNMPD.CONF File

The SNMPD.CONF file defines the SNMP agent security and notification destinations. If the SNMPD.CONF file exists, the agent can support SNMPv1, SNMPv2c, and SNMPv3 requests. If no SNMPD.CONF file exists, the agent will support only SNMPv1 and SNMPv2c requests.

Note: If the SNMPD.CONF file is found, the PW.SRC file and the SNMPTRAP.DEST files are not used.

SNMPD.CONF Dynamic Configuration: If the SNMPD.CONF information is located in an MVS data set rather than an HFS file, special considerations must be made to support dynamic configuration changes to the SNMP agent's configuration. If dynamic configuration changes are made, the file is rewritten to reflect the changes. Therefore, consider the following when allocating the SNMPD.CONF file to an MVS data set:

- The record length (LRECL) should be 512 bytes to accomodate the longest possible entry.
- The use of a member of a partitioned data set is tolerated but not recommended. Because the file might be rewritten often, frequent compression of the partitioned data set may become necessary. In addition, locking on the file is done at the data set level, not at the member level, so other members of the partitioned data set would not be usable while the SNMP agent was running (once a dynamic configuration change had been made).

SNMPD.CONF Example: A sample SNMPD.CONF file is shipped as /usr/lpp/tcpip/samples/snmpd.conf.

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

The sample OSNMP.CONF file used by the osnmp command contains entries that match the sample SNMPD.CONF data set. See "Configure the z/OS UNIX SNMP (osnmp) Command" on page 468 for additional information on configuring the osnmp command.

Note: By default, the SNMP agent and the osnmp command send packets such that a VIPA address will be used as the originating address in the packet, if SOURCEVIPA is configured. This is a change introduced in V2R10; previously, the SNMP agent and the osnmp command set a socket option to cause the physical interface addresses to be used as the originating addresses on packets they sent. That meant the SNMPD.CONF file had to contain all of the possible physical interface addresses that might be used, rather than a smaller number of VIPA addresses. A customer can override this change in behavior, if desired, by invoking the SNMP agent with the -a option or by using either the -a option or the NOSVIPA option in the osnmp command's OSNMP.CONF configuration file.

SNMPD.BOOTS

The SNMP agent uses the SNMPD.BOOTS configuration file to support SNMPv3 security. This file contains agent information used to authenticate the SNMPv3 requests. The SNMPD.BOOTS keeps the agent identifier and the number of times the agent reboots. If no SNMPD.BOOTS file exists when the agent is started, the agent creates one. You may want to add comments to the beginning of this file. If a file does exist, the agent uses the values specified in the file for setting its engineID and engineBoots values. If the file exists but contains invalid values for engineID or engineBoots, the agent issues a message and terminates.

Notes:

1. The recommended approach is to allow the SNMP agent to create the file.
2. If the SNMPD.BOOTS file is not provided, the SNMP agent creates the file. If multiple SNMPv3 agents are running on the same MVS image, use the

environment variable to specify different SNMPD.BOOTS files for the different agents. For security reasons, ensure unique engineIDs are used for different SNMP agents.

Creating User Keys

Authentication

Authentication is generally required for SNMPv3 requests to be processed (unless the security level requested is 'noAuth'). When authenticating a request, the SNMP agent verifies that the authentication key sent in an SNMPv3 request can be used to create a message digest that matches the message digest created from the authentication key defined for the user.

The osnmp command uses the authentication key found on an entry in the OSNMP.CONF configuration file. It needs to correlate with the authentication key specified on a USM_USER entry for that user in the agent's SNMPD.CONF configuration file.

As an alternative to storing authentication keys in the client configuration file, the osnmp command allows user passwords to be stored. If the osnmp command is configured with a password, the code generates an authentication key (and privacy key if requested) for the user. These keys must, of course, produce the same authentication values as the keys configured for the USM_USER in the agent's SNMPD.CONF file or configured dynamically with SNMP SET commands. However, the use of passwords in the client configuration file is considered less secure than the use of keys in the configuration file.

The authentication key is generated from two pieces of information:

- The specified password.
- The identification of the SNMP agent at which the key will be used. If the agent is an IBM agent and its engineID was generated using the vendor-specific engineID formula, the agent may be identified by IP address or host name. Otherwise, the engineID must be provided as the agent identification.

A key that incorporates the identification of the agent at which it will be used is called a localized key. It can be used only at that agent. A key that does not incorporate the engineID of the agent at which it will be used is called nonlocalized.

Keys stored in the osnmp command's configuration file, OSNMP.CONF, are expected to be nonlocalized keys. Keys stored in the SNMP agent's configuration file, SNMPD.CONF, can be either localized or nonlocalized, though the use of localized keys is considered more secure.

Encryption

Keys used for encryption are generated using the same algorithms as those used for authentication. However, key lengths may differ. For example, an HMAC-SHA authentication key is 20 bytes long, but a localized encryption key used with HMAC-SHA is only 16 bytes long.

z/OS CS provides a facility called *pwtokey* that enables conversion of passwords into localized and nonlocalized authentication and privacy keys. The pwtokey procedure takes as input a password and an identifier of the agent and generates authentication and privacy keys. Since the procedure used by the pwtokey facility is the same algorithm used by the osnmp command, the person configuring the

SNMP agent can generate appropriate authentication and privacy keys to put in the SNMPD.CONF file for a user, given a particular password and the IP address at which the agent will run.

Use the pwtokey command to convert passwords into authentication and privacy keys. Refer to *z/OS Communications Server: IP Configuration Reference* for the command syntax.

Provide MIB Object Configuration Information

An installation can set values for selected MIB objects by providing OSNMPD.DATA information. A sample of OSNMP.DATA is installed as HFS file /usr/lpp/tcpip/samples/osnmpd.data. Refer to *z/OS Communications Server: IP Configuration Reference* for syntax information. If no OSNMPD.DATA file is found, the defaults for these MIB objects are as follows:

Object	Default
dpiPathNameForUnixStream	The default is /tmp/dpi_socket.
sysDescr	If the environment variable HOSTNAME exists, its value is used. Otherwise, the default value identifies the z/OS system under which the agent is running. The maximum length of this object is 255 octets.
sysContact	″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.
sysLocation	″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.
sysName	″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.
sysObjectId	1.3.6.1.4.1.2.3.13
sysServices	A single octet that defaults to 0. See the RFC 1907 description for this object.
snmpEnableAuthenTraps	Default value is 2, which means traps are disabled.
saDefaultTimeout	5 seconds.
saMaxTimeOut	600 seconds.
saAllowDuplicateIDs	Default is 1, which means multiple instances of a subagent are allowed. **Note:** Because a subagent identifier cannot be specified for DPI version 1 subagents, a constant identifier is used for all version 1 subagents. Therefore, this object must be set to ″Yes″ (1) to allow multiple DPI version 1 subagents to run concurrently.

For information about where these MIB objects are defined, refer to the *z/OS Communications Server: IP User's Guide*.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

Start the SNMP Agent (OSNMPD)

The SNMP agent (OSNMPD) runs in a separate address space that executes load module EZASNMPD. OSNMPD can be started with or without parameters. When starting OSNMPD from MVS, add the parameters to the PARMS= keyword on the EXEC statement of the OSNMPD cataloged procedure. When starting OSNMPD from z/OS UNIX, specify the desired parameters on the osnmpd command. Refer to *z/OS Communications Server: IP Configuration Reference* for the command syntax.

Sample JCL Procedure for Starting OSNMPD from MVS

Update cataloged procedure OSNMPD by copying the sample in *hlq*.SEZAINST(OSNMPDPR) to your system or recognized PROCLIB. Change the data set names as required to suit your local configuration. The OSNMPD address space requires access to the IBM C/370 Library during execution.

Parameters may be passed to the agent on the `PARMS=` keyword on the EXEC statement of the OSNMPD cataloged procedure. Refer to *z/OS Communications Server: IP Configuration Reference* for the command syntax and parameter information. Any agent parameters you wish to specify may be added as shown in the following example:

```
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON)/ -c abc -d 255 -p 761'
```

In this example, the agent will use port 761 to accept requests, community name 'abc' will be added to the list of commnity names supported by the agent, and all agent traces will be activated. For more information on tracing, see the *z/OS Communications Server: IP Diagnosis*.

Starting OSNMPD from z/OS UNIX

To run the SNMP agent in background, you must add an ampersand (&) to the command and the issuer of the command must be in z/OS UNIX superuser mode. For a detailed explanation of the osnmpd parameters, refer to *z/OS Communications Server: IP Configuration Reference*.

Any agent parameters you wish to specify may be added as shown in the following example:

```
osnmpd -c abc -d 255 -p 761
```

In this example, the agent will use port 761 to accept requests, community name 'abc' will be added to the list of commnity names supported by the agent, and all agent traces will be activated. For more information on tracing, see *z/OS Communications Server: IP Diagnosis*.

Step 2: Configure the SNMP Commands

The two SNMP client applications provided with z/OS CS are:

- SNMP command from the NetView environment
- osnmp command in the z/OS shell

The SNMP command in the NetView environment requires the use of the NetView product. It supports SNMP version 1. The osnmp command in the z/OS shell

supports SNMP versions 1, 2, and 3. Depending on your requirements, you may decide to configure either or both of these clients, or to use an SNMP client on another platform.

Configure the NetView SNMP (SNMP) Command

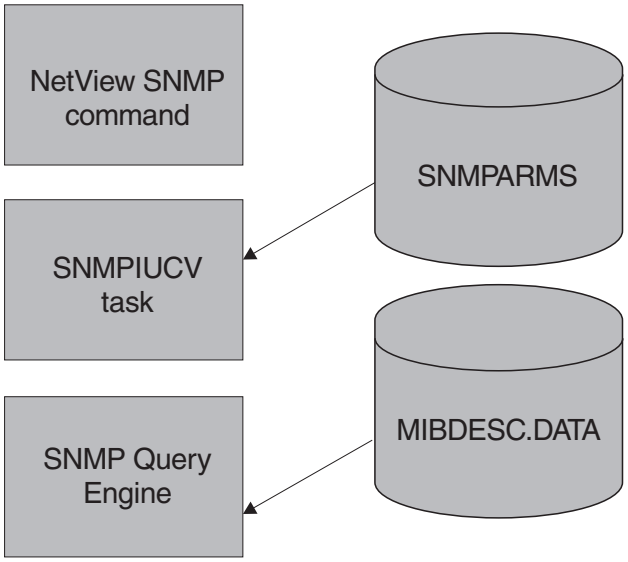


Figure 54. Configuration Files for NetView SNMP

The SNMP command in the NetView environment can be used to send SNMP version 1 requests to SNMP agents on either local or remote hosts. The SNMP command requires the command processor itself, the SNMPIUCV task for inter-address space communication, and the SNMP query engine, which creates the packets sent to the SNMP agent. The NetView SNMP command supports only community-based security.

Configure the SNMP Query Engine

Update the SNMPQE cataloged procedure by copying the sample in *hlq*.SEZAINST(SNMPPROC) to your system or recognized PROCLIB. Specify SNMP parameters and change the data set names as required to suit your local configuration. The SNMPQE address space requires access to the IBM C/370 Library during execution.

The SNMP query engine (SQESERV) needs access to the *hlq*.MIBDESC.DATA data set for the MIB variable descriptions. You can find a sample of this data set in *hlq*.SEZAINST(MIBDESC).

MIBDESC.DATA Data Set: The MIBDESC.DATA data set defines the short names for MIB variables. Short names are the character representation for the ASN.1 variable names. For example, sysUpTime is the short name for 1.3.6.1.2.1.1.3.0 (the MIB variable that stores the time since the SNMP agent was last restarted). Short names are generally shown as a combination of upper and lowercase characters, though SNMP on z/OS CS ignores these case distinctions. Variable names must always be in ASN.1 language when they are sent to an SNMP agent. You can always use ASN.1 language to specify the variable names in an enterprise-specific tree (assuming that the agent supports them). You can use these short names to specify the MIB variables.

When you issue an SNMP GET, GETNEXT, or SET command, and specify the variable name in ASN.1 notation, the SNMP Query Engine uses that name and sends it in the SNMP packet to the agent. When you issue an SNMP GET, GETNEXT, or SET command, and specify the short name for the variable (for example, sysDescr), the SNMP Query Engine looks for that name in the MIBDESC.DATA data set and uses the ASN.1 name specified in the data set when it sends the SNMP packet to the agent.

The SNMPQE address space must be able to access the MIBDESC.DATA data set.

You can change the short names in the MIBDESC.DATA data set to the equivalent in your national language. You can also leave the current names and add the equivalent names in your national language. However, the SNMP MIBVNAME function returns only the first entry found in the data set that satisfies the search. In addition, all enterprise-specific variables used by hosts in your network should be added to this data set.

Entries in the data set do not need to be in a specific sequence. Each name starts on a new line. The following shows the line format.

```
short_name asn.1_name type time_to_live
```

Each variable on the line is separated by either one or more spaces or tabs. An asterisk (*) in column 1 indicates that the line is a comment line.

Following is a sample MIBDESC.DATA line with a sysDescr variable translated in Dutch and a few enterprise variables added (in this example, company ABC received 1.3.6.1.4.1.42 as the ASN.1 number for their enterprise):

```
*-------------------------------------------------------------*
* MIB Variable name | ASN.1 notation        | Type     | TTL  *
*-------------------------------------------------------------*
* Following is Dutch name for sysDescr
systeemBeschrijving  1.3.6.1.2.1.1.1.         display    900
sysDescr             1.3.6.1.2.1.1.1.         display    900
  ...
  other entries
  ...
* Following are enterprise specific variables for company ABC
ABCInfoPhone         1.3.6.1.4.1.42.1.1       display    900
ABCInfoAddress       1.3.6.1.4.1.42.1.2       display    900
```

The TTL field contains the number of seconds that a variable lives in the Query Engine's internal cache. If there are multiple requests for the same variable within the TTL period, the variable value is obtained from the cache, and unnecessary network traffic is avoided.

You can define multiple short names or text names for the same variable, as shown with the Dutch translation of the sysDescr variable. In this case, the SNMP Query Engine returns the first value in the table on an SNMP MIBVNAME request. In the previous example, the SNMP Query Engine would return `systeemBeschrijving` and not `sysDescr`. The name returned is in mixed case.

When the SNMP Query Engine receives a short name or text name in a GET, GETNEXT, or SET request, it compares the name against the entries in the MIBDESC.DATA data set. This comparison is not case-sensitive. For example, a request for `SYSDESCR`, `SysDescr`, or `sysDescr` matches the `sysDescr` entry with an ASN.1 notation of `1.3.6.1.2.1.1.1.`.

When the SNMP Query Engine receives an SNMP response, it looks up the variable in the MIBDESC.DATA table Type field for information about translating the value into displayable characters. The information contained in the Type field is case-sensitive and must be specified in lowercase.

Note: If you are using SNMP to receive response or trap PDUs which contain enterprise specific variables, the variables must be added to the MIBDESC.DATA data set.

Specifying the SNMPQE Parameters: The SQESERV module can be configured to start without parameters or you can add any of the following parameters to `PARMS='` in the PROC statement of the SNMPQE cataloged procedure. For example,

```
//SNMPQE   PROC MODULE=SNMPQE,PARMS='-h MVSA'
```

Refer to *z/OS Communications Server: IP Configuration Reference* for the command syntax.

Refer to *z/OS Communications Server: IP Diagnosis* for more information on tracing.

Setting Up Authorization for SNMPQE: To create RAW sockets necessary for SNMP PING requests, the user ID associated with the SNMPQE started task must have superuser authority (z/OS UNIX UID of 0) or be permitted to BPX.SUPERUSER facility authority.

Configure NetView as an SNMP Monitor

To configure the NetView interface as an SNMP monitor, perform each of the following tasks:

- Configure for SNMPIUCV
- Configure for the SNMP command processor
- Configure for the SNMP messages
- Update the SNMP initialization parameters

Configure for SNMPIUCV: SNMPIUCV is the NetView optional task that handles IUCV communication between the NetView program and the SNMP query engine. SNMPIUCV resides in the *hlq*.SEZADSIL data set.

Add the following TASK statement for SNMPIUCV to the DSIDMN member of the data set specified by the DSIPARM DD statement in the NetView start procedure.

```
TASK  MOD=SNMPIUCV,TSKID=SNMPIUCV,PRI=5,INIT=Y
```

This statement causes SNMPIUCV to start automatically when the NetView program is started.

If you specify `INIT=N` instead of `INIT=Y` in the TASK statement for SNMPIUCV, a NetView operator can start the SNMPIUCV task by entering the following:

```
START TASK=SNMPIUCV
```

The SNMPIUCV task tries to connect through IUCV to the SNMP query engine. If this fails, it retries the connect as specified by the SNMPQERT keyword in the SNMPARMS member of the *hlq*.SEZADSIP data set. The default is every 60 seconds.

Configure for the SNMP Command Processor: SNMP is the command processor that allows NetView operators and CLISTs to issue SNMP commands.

SNMP resides in the *hlq*.SEZADSIL data set. This data set should be concatenated to the STEPLIB DD statement in the NetView start procedure.

Add the following statement to the DSICMD member of the data set specified by the DSIPARM DD statement in the NetView start procedure.

```
SNMP  CMDMDL  MOD=SNMP,ECHO=Y,TYPE=R,RES=Y
```

After the SNMPIUCV task is started, you can issue the SNMP command. The SNMP command passes a request to the SNMPIUCV task to forward to SNMPQE. The return code represents a request number that is associated with the request. The responses are returned asynchronously and contain this request number. The operator or CLIST must use the request number to correlate the response to the request.

Configure for the SNMP Messages: The NetView SNMP messages reside in the *hlq*.SEZADSIM data set as DSISNM*nn*, where *nn* is the number of the member. The valid message members are DSISNM00 through DSISNM05, DSISNM10, DSISNM12, and DSISNM99. The data set containing these members should be added to the DSIMSG DD statement in the NetView start procedure.

Update the SNMP Initialization Parameters: SNMPIUCV reads the SNMPARMS member in the *hlq*.SEZADSIP data set at startup. This data set contains the initialization parameters for SNMP. The data set containing SNMPARMS should be added to the DSIPARM DD statement in the NetView startup procedure. Refer to *z/OS Communications Server: IP Configuration Reference* for detailed information for the SNMP parameter data set (SNMPARMS).

Configure the z/OS UNIX SNMP (osnmp) Command

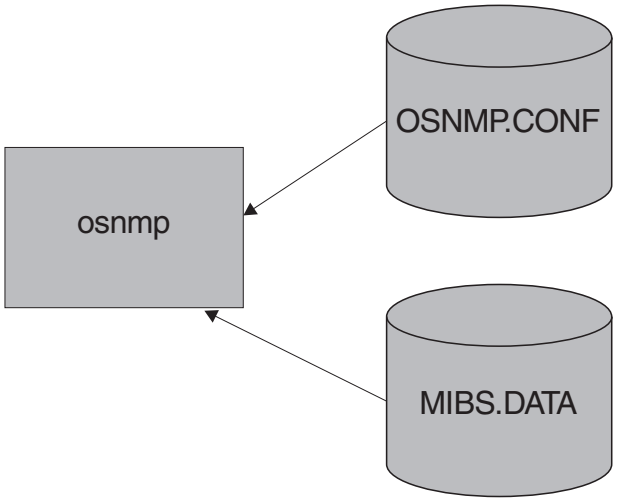


Figure 55. Configuration Files for osnmp

The osnmp command is used to send SNMP requests to SNMP agents on local or remote hosts. The requests can be SNMPv1, SNMPv2, or SNMPv3. For SNMPv2 and SNMPv3 requests, the OSNMP.CONF configuration file is required. The *winSNMPname* specified on an OSNMP.CONF statement can be used as the value of the -h parameter on the osnmp command. For a detailed explanation of the parameters you can specify on the osnmp command, see the *z/OS Communications Server: IP User's Guide*.

To configure the osnmp command, perform the following tasks:

- Provide osnmp configuration information
- Provide user MIB object information

Provide osnmp Configuration Information

The OSNMP.CONF file is used to define target agents and, for SNMPv3, the security parameters to be used in sending requests to them.

The contents of the file, regardless of location, are the same. Only the first file found is used. A sample of this file is installed as HFS file /usr/lpp/tcpip/samples/snmpv2.conf. This sample should be copied and modified for your installation. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Examples:

- Example 1:

  The following entry defines an SNMPv2c node for osnmp:

  ```
  mvs1     9.67.113.79 snmpv2c
  ```

  where *mvs1* is the name used with the -h parameter on the osnmp command and *9.67.113.79* is the IP address of the SNMPv2c agent.

- Example 2:

  The following entry defines an SNMPv3 node:
  where *v3mak* is the name used on the -h parameter of the osnmp command.

```
v3mak   127.0.0.1 snmpv3 u1 - - AuthNoPriv  HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987 - -
```

  SNMP requests sent using this entry uses USM user name *u1* using HMAC-MD5 authentication but no encryption.

- Example 3:

  The following entry defines an SNMPv3 node. The needed authentication and privacy keys will be generated from the password *u6password*.
  The USM user is *u6*. The authentication protocol is HMAC-SHA, and CBC 56-bit

```
v3sap   127.0.0.1 snmpv3 u6 u6password - AuthNoPriv   HMAC-SHA -   DES -
```

  DES encryption is used.

Provide User MIB Object Information

If you want to use the textual names for MIB objects that are not defined in the compiled MIB, then you can define them to the osnmp command using the MIBS.DATA file. All the objects in the list in the “Management Information Base (MIB) Objects” appendix in the *z/OS Communications Server: IP User’s Guide* are in the compiled MIB. A sample of the MIBS.DATA file is installed as HFS file /usr/lpp/tcpip/samples/mibs.data. Copy this sample and modify it for your installation.

MIBS.DATA Statement Syntax

The MIBS.DATA statements syntax can be used to specify character (usually called ’textual’) names for MIB objects not defined in any compiled MIB supplied with z/OS CS. You can then use these character/textual names as the name of the objects on the osnmp command.

The format of a statement in this file is:

```
character_object_name object_identifier object_type
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Step 3: Configure the SNMP Subagents

This section contains information about configuring the TCP/IP subagent.

There are three SNMP subagents shipped with z/OS CS:

- The TCP/IP subagent reports information about the TCP/IP stack.
- The OMPROUTE subagent reports information specific to OSPF. The ROUTESA_CONFIG statement is used in the OMPROUTE configuration file to configure the OMPROUTE subagent. For details on ROUTESA_CONFIG, refer to *z/OS Communications Server: IP Configuration Reference*.
- The SLA subagent reports information about defined service policies and performance statistics related to traffic using those policies. For configuration information about the SLA subagent, see "Chapter 9. Policy-Based Networking" on page 411.

There are two statements in the profile data set used to configure the TCP/IP subagent, the SACONFIG and ITRACE statements.

- SACONFIG

  Use the SACONFIG statement to configure the subagent. The SACONFIG parameters determine whether or not the subagent is automatically started at TCP/IP initialization, what port number to use to contact the agent, and other configuration values. For a detailed explanation of this statement, refer to *z/OS Communications Server: IP Configuration Reference*.
- ITRACE

  Use the ITRACE statement to determine what trace information, if any, should be recorded by the subagent. For a detailed explanation of this statement, refer to *z/OS Communications Server: IP Configuration Reference*.

Step 4: Configure the Open Systems Adapter (OSA) Support

The TCP/IP subagent can retrieve ATM and Ethernet management data from the Open Systems Adapter Support Facility (OSA/SF) for several OSA adapters. The ATM management data is supported for any ATM OSA or OSA-Express ATM155 adapters. The Ethernet management data is supported for any OSA-Express Gigabit Ethernet or OSA-Express QDIO Fast Ethernet adapters.

In order to obtain the management data, the adapters must be defined to the TCP/IP stack where the subagent is active, via DEVICE/LINK statements in the TCP/IP Profile. To retrieve ATM management data, the ATM adapter must be defined as an ATM device/link, even if it is configured for ATM LAN emulation mode and is therefore also defined to some TCP/IP instance as an LCS or MPCIPA device. For OSA-Express ATM155 adapters configured for QDIO LAN Emulation mode, you can use one of the adapter's logical portnames on the PORTNAME parameter of the ATM DEVICE statement. To retrieve Ethernet management data, the OSA-Express adapter must be defined as an MPCIPA Ethernet link. If the portname is manually configured at the adapter, then management data can be retrieved from the adapter even if it is not active and not in use by any TCP/IP stack or by VTAM. If the portname is dynamically configured (e.g. MPCIPA links), then the adapter has to be active to some TCP/IP stack to retrieve the management data.

Current support consists of:

- Interface table from RFC2233 for ATM devices, Native and IP Forwarding links, LAN emulation links, AAL5 and ATM layer interfaces
- ATM Channel, Port and PVC tables from the IBM MVS Enterprise Specfic MIB
- ATM LAN Emulation tables from the IBM MVS Enterprise Specific MIB
- atmInterfaceConfTable from RFC1695
- dot3StatsTable from EtherLike-MIB in RFC2665
- Asynchronous SNMP Trap generation for operational management:
  - ATM Port enabled - LinkUp Trap
  - ATM Port disabled - LinkDown Trap
  - ATM Permanent Virtual Circuit (PVC) creation - ibmMvsAtmOsasfAtmPvcCreate Trap (ATM OSA only)
  - ATM Permanent Virtual Circuit (PVC) deletion - ibmMvsAtmOsasfAtmPvcDelete Trap (ATM OSA only)
- Provide method for assigning an IP Address to the ATM Port

  Use the *osnmp set* command against the ibmMvsAtmOsasfPortIpAddress MIB object to assign the IP Address.

OSA/SF Prerequisites

The TCP/IP subagent provided by z/OS CS will connect to OSA/SF to obtain management data. For a subagent to establish a connection to OSA/SF, two OSA/SF components must be started:

- IOAOSASF

  IOAOSASF is a sample JCL procedure that can be used to start the main OSA/SF address space. The sample has a job name of OSASF1.
- IOASNMP

  IOASNMP is a sample JCL procedure that starts the OSA/SF-provided z/OS UNIX transport application that interconnects a subagent with OSASF1.

These sample procedures and all entities that they call are provided with OSA/SF. For a detailed explanation of how to set up OSA/SF on your MVS system, see GC23-3870. The primary purpose of OSA/SF is to manage OSA Adapters. It has been extended to support OSA management via SNMP. An instance of IOAOSASF, IOASNMP, TCPIP and its subagent, and an SNMP agent must be started on every MVS image where OSA management support is needed.

The recommended startup order is:

1. Start IOAOSASF and wait until it completely initializes. IOAOSASF must be started before IOASNMP.
2. Start TCP/IP and wait until TCP/IP completes its initialization. Actually IOAOSASF can be started after TCP/IP but must be started prior to the next step.
3. Start IOASNMP after TCP/IP is initialized. If starting multiple TCP/IP instances that run under z/OS UNIX as AF_INET Physical File Systems, wait until at least one TCP/IP where OSA/SF support was requested has initialized. OSA/SF support is requested by specifying the OSASF parameter on the SACONFIG statement in the profile data set for a TCP/IP instance. For a detailed description of the SACONFIG statement, refer to *z/OS Communications Server: IP Configuration Reference*.

4. Start the SNMP Agent, OSNMPD, for each TCP/IP instance where OSA management support is desired.

On an MVS image only a single instance of either IOAOSASF or IOASNMP can (or should) be started. An attempt to start multiple copies of IOAOSASF will be rejected. Starting multiple copies of IOASNMP will yield unpredictable results.

Ensure that OSA/SF is at Version 2 Release 1 level or higher with the OSA/SF APARs OW39984 and OW42352 applied.

Required TCP/IP Profile Statements

For a detailed description of the statements mentioned here, see refer to *z/OS Communications Server: IP Configuration Reference*. The following TCP/IP profile statements must be updated for OSA management support:

- SACONFIG

  On the SACONFIG statement, OSA Management support must be enabled by specifying OSAENABLED. Omission of OSAENABLED when TCP/IP is started will result in no OSA management support. The SACONFIG statement controls the operation of the subagent that runs in a TCP/IP address space as a separate task.

  The OSASF parameter specifies which port IOASNMP should use to Listen for connections from subagents to OSA/SF. For an explanation of the usage of this parameter when starting multiple TCP/IP instances, see “Multiple TCP/IP Instances Considerations” on page 473. It is recommended that the OSASF port be reserved by also specifying it on a PORT statement.

  For example:

  ```
  SACONFIG OSAENABLED OSASF 721
  ```

- PORT

  Prereserve the port to be used in communication with OSA/SF.

  For example:

  ```
  PORT
     721  IOASNMP ; OSA/SF TCP/IP Communications
  ```

  In the above example since IOASNMP runs as an z/OS UNIX application the port could have been reserved for z/OS UNIX. Review the /etc/services HFS file to insure that there are no port conflicts.

- DEVICE and LINK

  Provide ATM DEVICE and LINK statements for any OSA ATM adapter for which you want SNMP ATM management data. For example:

  ```
  DEVICE osaName ATM PORTNAME portname
  LINK linkName ATM osaName
  ```

  Provide DEVICE and LINK statements for any OSA-Express Ethernet adapter for which you want SNMP Ethernet management data. For example:

  ```
  DEVICE portname MPCIPA NONROUTER
  LINK linkName IPAQENET portname
  ```

Multiple TCP/IP Instances Considerations

Subagent Connection to OSA/SF

When multiple z/OS CS instances are active in the same MVS image, only one of the TCP/IP instances is connected to IOASNMP. In order for a TCP/IP instance to connect to IOASNMP the OSASF parameter must be specified on the SACONFIG Profile statement.

IOASNMP connects to a TCP/IP instance and acts as a server, receiving connections from those TCP/IP subagents where OSAENABLED was specified on the SACONFIG Profile statement. The result is that all these subagents connect through the same TCP/IP to IOASNMP in order to retrieve OSA information from OSA/SF. For a depiction of this process, see Figure 56.

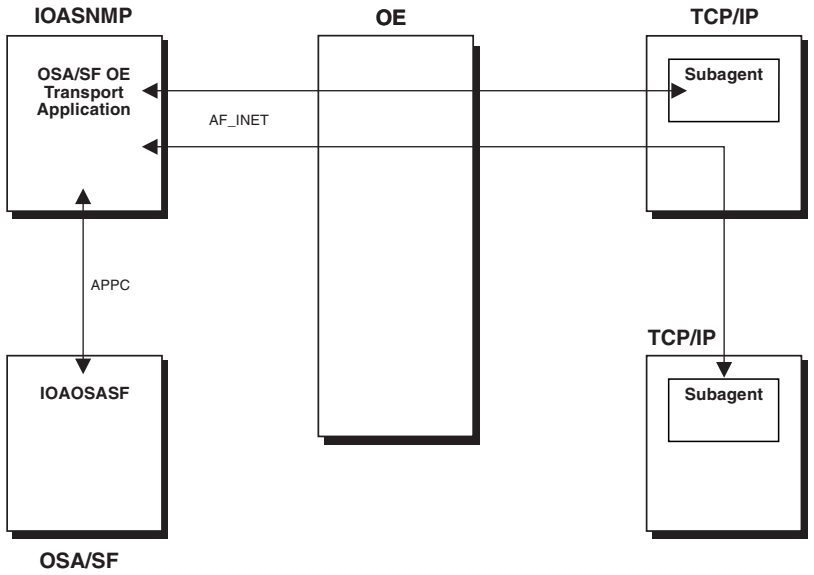


Figure 56. Subagent Connection to OSA/SF

If IOASNMP loses its connection to TCP/IP it terminates and needs to be restarted.

If the currently connected TCP/IP terminates, IOASNMP will attempt to connect to another TCP/IP instance for which the OSASF parameter was specified on the SACONFIG Profile statement, using the port number specified for the OSASF parameter. The subagents will also attempt to reconnect to OSA/SF via IOASNMP using this same OSASF port number. For this reason it is recommended that the same OSASF port number be used on the SACONFIG statement of every TCP/IP instance where the OSASF parameter is specified.

Whenever a socket error occurs on the OSA/SF socket, the connected subagents will issue the following message:

```
EZZ3219I SNMP SUBAGENT: DISCONNECTED FROM OSA/SF
```

When the subagent connection is reestablished, the following message is issued:

```
EZZ3218I SNMP SUBAGENT: CONNECTED TO OSA/SF
```

Step 5: Configure the Trap Forwarder Daemon

Most SNMPv1 agents forward traps to port 162. If a manager needs to listen for these traps, it has to bind to port 162 and listen for it. When a manager has already issued a bind it is impossible for another manager to listen for the same traps. The Trap Forwarder daemon eliminates this problem by listening for traps on port 162 and forwarding them to all the configured managers.

You can configure the Trap Forwarder daemon to receive a trap on a specified port and forward it to multiple other ports on the same host and on different hosts. This will allow multiple SNMP managers on an z/OS to be able to receive all the traps sent to one port.

To configure the Trap Forwarder daemon, perform the following tasks:

1. Provide PROFILE.TCPIP statements.
2. Provide Trap Forwarder configuration.
3. Start the Trap Forwarder daemon (TRAPFWD).

Provide PROFILE.TCPIP Statements

Update the PORT following configuration statements in hlq.PROFILE.TCPIP.

The default port used by the trap forwarder daemon to receive trap datagrams is UDP port 162. If you want to ensure that no other application uses this port, you must specify the following PORT statement:

```
PORT
  162 UDP TRAPFWD ;  Trap Forwarder daemon
```

If the daemon will be started from the z/OS shell, reserve the port for z/OS UNIX by changing OMVS instead of TRAPFWD. Note that by doing so, the osnmp command could make use of the port if it is started (with the trap option) before TRAPFWD is started.

Provide Trap Forwarder Configuration Information

The TRAPFWD.CONF file defines the configuration parameters for trapfwd daemon.

A sample of the TRAPFWD.CONF is shown below:

```
#
# A sample configuration file for trapfwd
#
# Syntax  : ip_address/host_name            port_number            option
#
#
9.67.113.79               1064           ADD_RECVFROM_INFO
myHost                    1066
myHost                    1067           ADD_RECVFROM_INFO
myHost                    1099
9.67.35.37                1064           ADD_RECVFROM_INFO
-                         1065
-                         1068           ADD_RECVFROM_INFO
```

For more information about the statement syntax, refer to *z/OS Communications Server: IP Configuration Reference*.

Starting and Stopping the Trap Forwarder Daemon

The Trap Forwarder daemon can be started from the z/OS UNIX shell or from the MVS console.

Starting the Trap Forwarder daemon from an z/OS UNIX

This example starts the Trap Forwarder daemon on the standard port (port 162).

```
# trapfwd
```

This example starts the Trap Forwarder daemon on a particular port (port 5062).

```
# trapfwd -p 5062
```

Starting the trap forwarder daemon from an MVS console: Update cataloged procedure TRAPFWD by copying the sample in hlq.SEZAINST(EZASNTRA) to your system. Refer to *z/OS Communications Server: IP Configuration Reference* the for more information about this cataloged procedure.

Stopping the trap forwarder daemon: From MVS:

```
P TRAPFWD
```

If TRAPFWD was started from a cataloged procedure, procname is the member name of that procedure. If TRAPFWD was started from the z/OS shell, procname is useridX, where X is the sequence number set by the system. To determine the sequence number issue d omvs u=userid from the console. This will show the programs running under the user ID.

From UNIX:

```
kill < pid >
```

Issue the kill command to the process ID (PID) associated with TRAPFWD. To find the PID issue the ″ps -ef″ command from the z/OS shell.

Tracing: The modify command can be used to display the existing tracing level and also to change the tracing level.

The following example sets the trace level of the Trap Forwarder Daemon to 1.

```
MODIFY TRAPFWD,TRACE,LEVEL=1
```

The following example displays the level of tracing set for the Trap Forwarder Daemon.

```
MODIFY TRAPFWD,TRACE,QUERY
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Dynamically Refreshing Configuration: The modify command can be used to dynamically refresh the configuration information. When this is done, the old configuration information is discarded completely. The configuration file is read again and the daemon is initialized.

The following example refreshes the configuration by reading the configuration file.

```
MODIFY TRAPFWD,REFRESH
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Chapter 11. Remote Print Server (LPD)

Read “Understanding Search Orders of Configuration Information” on page 6. It covers important information about data set naming and search sequences.

The Remote Print (LPD) server supports the Line Print Daemon and allows you to print on JES controlled printers from any host in your TCP/IP network that implements the Line Print client functions. These client functions are invoked with the LPR command. LPR is available as a TSO command, and the LPD server is implemented as a started z/OS task.

Refer to the *z/OS Communications Server: IP Configuration Reference* for information on starting and stopping the TCP/IP print server (LPD). When LPD is stopped by the MVS operator with the *P procname* command, LPD will terminate any TCP/IP connections currently transferring data. Before ending, LPD will finish spooling to JES any print jobs that it has received and is currently spooling. JES will handle these jobs after LPD ends.

This chapter describes how to configure the LPD server. To operate the LPD server after it is running, refer to *z/OS Communications Server: IP Configuration Reference*.

The (SMTP or LPD) server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E will be issued and the server will terminate.

Configuring the Remote Print Server

Steps to configure the Remote Print Server:

1. Specify AUTOLOG and PORT statements in PROFILE.TCPIP.
2. Update the LPD server cataloged procedure.
3. Update the LPD server configuration data set.
4. Create a banner page (optional).

For information about operating and controlling the LPD server, refer to *z/OS Communications Server: IP Configuration Reference*.

Step 1: Configuring PROFILE.TCPIP for LPD

If you want the LPD server started automatically when the TCPIP address space is started, include the name of the member containing the LPD server cataloged procedure in the AUTOLOG list in *hlq*.PROFILE.TCPIP. For example:

```
AUTOLOG
  LPSERVE
ENDAUTOLOG
```

To ensure that port TCP 515 is reserved for the LPD server, also add the name of the member containing the LPD cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP. For example:

```
PORT
  515 TCP LPSERVE
```

See the *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Updating the LPD Server Cataloged Procedure

Update the LPD server cataloged procedure to suit your local configuration by copying the sample to your system or recognized PROCLIB from *hlq*.SEZAINST(LPSPROC) and modifying the sample. Specify LPD parameters, and change the data set names as required. See "Specifying LPD Server Parameters" for more information on the LPD server parameters.

Specifying LPD Server Parameters

The system parameters required by the LPD server are passed by the PARM option on the EXEC statement of the LPD cataloged procedure. Update the following parameters as required.

LPDDATA='*data_set_name***'**
Specifies the fully qualified name of the data set containing the LPD configuration statements. This data set can be sequential or a member of a PDS.

LPDPRFX='PREFIX *your_prefix***'**
Specifies the high-level qualifier to be used for temporary data sets created by the LPD server. Include both the PREFIX keyword and your qualifier in the quoted string. The qualifier may be up to 26 characters. If it is blank, it defaults to the procedure name. The LPD task requires the authority to create and modify data sets with this prefix.

DIAG='*options***'**
Specifies any of the following diagnostic options in a quoted string of keywords separated by blanks. For example, `DIAG='VERSION TRACE'`

VERSION
Displays the version number.

TYPE Activates high-level trace facility in the LPD server. Significant events, such as the receipt of a job for printing, are recorded in the //SYSOUT DD data set specified in your LPD server cataloged procedure.

TRACE
Causes a detailed trace of activities within the LPD server to record in the //SYSOUT DD data set specified in your LPD server cataloged procedure. The detailed tracing can be also activated with the DEBUG statement in the LPD server configuration data set and with the TRACE command of the SMSG interface.

Note: The JCL PARM= statement has a limit of 100 characters.

Configuring LPDDATA

Use a DD card that requires the LPD configuration file to be in a data set.

```
//SYSLPDD  DD DISP=SHR,DSN=TCPIP.V2R7.SEZAINST(LPDDATA)
```

Using this example, the DD card must be specified as SYSLPDD, but the data set name can be any valid data set name with a member specified up to 44 characters.

In order to use the DD card method, you **must** comment out the LPDDATA= and remove ″&LPDDATA″; from the PARM= statement.

Note: The search order for the configuration file is:
1. LPDDATA= on the PARM= statement
2. //SYSLPDD DD statement
3. *hlq*.LPD.CONFIG

If both the LPDDATA= statement on the PARM= statement and the //SYSLPDD DD statement are specified, the data set name specified on LPDDATA= is used.

The LPD server does not limit the number of print jobs it handles per connection. This can cause a memory abend to occur if too many print jobs are sent in one connection. Certain LPR clients, such as SUN UNIX, are set up to send multiple jobs in one connection. It is recommended that the LPD start procedure be started with a region size of 6M and the LPR client send no more than 50 print jobs in one connection.

Step 3: Updating the LPD Server Configuration Data Set

The LPD configuration data set defines the local, NJE, and remote services (printers and punches) used by the LPD server. To update the LPD server configuration data set, copy the sample provided in *hlq*.SEZAINST(LPDDATA) and modify it to suit your installation. Refer to *z/OS Communications Server: IP Configuration Reference* for more details.

- To define a printer or punch:
  - Include a SERVICE statement with the appropriate parameters for each printer or punch your are defining.
  - Specify the type of service with the LOCAL, NJE, or REMOTE parameter in the SERVICE statement.
  - For local or NJE services, include any of the optional parameters to further define the service: EXIT, FAILEDJOB, FILTERS, LINESIZE, PAGESIZE, RACF, and SMTP.
  - For remote services, specify the destination printer and host. Any additional specifications are defined on the remote system and are not necessary in the SERVICE statement.
- To turn on LPD server tracing, include the optional DEBUG statement.
- To authorize users for the SMSG interface, include the optional OBEY statement.
- Printer names cannot contain an at sign (@).

Step 4: Creating a Banner Page (Optional)

LPBANNER is the name of the default program that is provided in executable form in the SEZATCP data set. This program is specified on the EXIT parameter in the SERVICE statement. LPBANNER prints a separator page that contains, in large letters, a banner stating “LPD BANNER”, the user ID, the job name, and the job class. Field headings of HOST, USER, JOB, and CLASS appear in smaller letters.

The sample exit LPBANNER uses machine carriage control and is designed to be used with the SERVICE PRINTER LOCAL or SERVICE PRINTER NJE statements. Banner pages are usually not used with the SERVICE PUNCH or SERVICE RECFMU statements; however, if you want to have banner pages (headers) for the SERVICE PUNCH or SERVICE RECFMU devices, a user created banner exit is required.

You can either use the executable form or copy and modify the sample source provided in *hlq*.SEZAINST(EZAAE04S) and *hlq*.SEZAINST(EZAAE04T) to create a banner. If you are changing the source to create your own banner, assemble and link-edit these data sets as reentrant. You can modify and use the following JCL to do this. Changing the ENTRY and NAME to something other than LPBANNER will avoid possible maintenance problems in the future.

```
//ASMLNK JOB MSGLEVEL=(1,1),MSGCLASS=A,CLASS=A,REGION=1024K
//ASM1   EXEC PGM=ASMA90,PARM='OBJECT,XREF(FULL)'
//STEPLIB DD  DISP=SHR,DSN=HLA.OSV1R4.SASMMOD1
//*  ASSEMBLER H
//SYSLIB   DD DSN=tcpip..V3R4.SEZACMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DSN=&&SYSUT1
//SYSPUNCH DD DUMMY,DCB=BLKSIZE=80
//SYSPRINT DD SYSOUT=A
//SYSLIN   DD DSN=&&OBJECT(EZAAE04S),DISP=(,PASS),UNIT=SYSDA,
//            SPACE=(CYL,(5,5,1)),DCB=BLKSIZE=400
//SYSIN    DD DSN=tcpip.V3R4.SEZAINST(EZAAE04S),DISP=SHR
/*
//ASM2   EXEC PGM=ASMA90,PARM='OBJECT,NODECK,XREF'
//STEPLIB  DD  DISP=SHR,DSN=HLA.OSV1R4.SASMMOD1
//*  ASSEMBLER H
//SYSLIB   DD DSN=tcpip.V3R4.SEZACMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DSN=&&SYSUT1
//SYSPUNCH DD DUMMY,DCB=BLKSIZE=80
//SYSPRINT DD SYSOUT=A
//SYSLIN   DD DSN=&&OBJECT(EZAAE04T),DISP=(OLD,PASS)
//SYSIN    DD DSN=tcpip.V3R4.SEZAINST(EZAAE04T),DISP=SHR
/*
//LNK    EXEC PGM=IEWL,PARM='LIST,NCAL,RENT,LET'
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD DSN=tcpip.V3R4.SEZATCP,DISP=SHR
//AEZAMOD1 DD DSN=tcpip.V3R4.AEZAMOD1,DISP=SHR
//OBJECT   DD DSN=&&OBJECT,DISP=(OLD,DELETE)
//SYSLIN   DD *
  ORDER   EZBOECPR
  INCLUDE AEZAMOD1(EZBOECPR)
  INCLUDE OBJECT(EZAAE04S)
  INCLUDE OBJECT(EZAAE04T)
  MODE    AMODE(24),RMODE(24)
  ENTRY   LPBANNER
  NAME    LPBANNER(R)
/*
```

Chapter 12. Remote Procedure Calls

This chapter contains information to help you configure:

- PORTMAP
- UNIX PORTMAP
- NCS
- NDB

Read “Understanding Search Orders of Configuration Information” on page 6. It covers important information about data set naming and search sequences.

Configuring the PORTMAP Address Space

This section describes how to configure the PORTMAP address spaces, which runs the Portmapper function.

Portmapper is the software that supplies client programs with the port numbers of server programs. Clients contact server programs by sending messages to port numbers where receiving processes receive the message. Because you make requests to the port number of a server rather than directly to a server program, client programs need a way to find the port number of the server programs they wish to call. Portmapper standardizes the way clients locate the port number of the server programs supported on a network.

Steps to configure PORTMAP:

1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the PORTMAP cataloged procedure.
3. Define the data set for well-known procedure names.

Step 1: Configuring PROFILE.TCPIP for PORTMAP

If you want the PORTMAP server to start automatically when the TCPIP address space is started, you should include PORTMAP in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  PORTMAP
ENDAUTOLOG
```

Note: If your system is using the Network File System (NFS) server, you ***must*** start the PORTMAP address space. See *z/OS Network File System Customization and Operation* for more information.

To ensure that port UDP 111 and TCP 111 are reserved for the PORTMAP server, also add the name of the member containing the PORTMAP cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  111 UDP PORTMAP
  111 TCP PORTMAP
```

See the *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Updating the PORTMAP Cataloged Procedure

Update the PORTMAP cataloged procedure to suit your local conditions by copying the sample provided in *hlq*.SEZAINST(PORTPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify PORTMAP parameters, and change the data set names, as required. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details.

Step 3: Defining the Data Set for Well-Known Procedure Names

z/OS CS includes a data set that contains a list of commonly used procedure names. This data set is used by the RPCINFO command to resolve remote program numbers into their equivalent program names.

To create the data set, copy the ETCRPC member of *hlq*.SEZAINST to the default data set called *hlq*.ETC.RPC. If a user has a *user_id*.ETC.RPC data set defined, it takes precedence over the preceding data set.

Normally, you would not change this data set except to add a new application to the list. To add a new application, add a line that contains the following items:

- The *program_name* of the new application or procedure
- The *program_number* of the new application or procedure
- Any comments regarding the description of the program

The items are variable in format, each separated by a blank.

The *hlq*.SEZAINST(ETCRPC) data set contains the well-known procedure names. Following is the ETCRPC sample.

```
#
#   rpc   1.2   86/10/07
#
#   COPYRIGHT = NONE.
#
portmapper    100000    portmap sunrpc
rstatd        100001    rstat rup perfmeter
rusersd       100002    rusers
nfs           100003    nfsprog
ypserv        100004    ypprog
mountd        100005    mount showmount
ypbind        100007
walld         100008    rwall shutdown
yppasswdd     100009    yppasswd
etherstatd    100010    etherstat
rquotad       100011    rquotaprog quota rquota
sprayd        100012    spray
3270_mapper   100013
rje_mapper    100014
selection_svc 100015    selnsvc
database_svc  100016
rexd          100017    rex
alis          100018
sched         100019
llockmgr      100020
nlockmgr      100021
x25.inr       100022
statmon       100023
status        100024
#
# NDB Program numbers added for more useful rpcinfo output
# Values are in decimal. Corresponding hexadecimal values
# for the NDB port manager is X'20000020' and for the NDB
# servers are from X'20000021' to X'20000084'.
#
```

```
ndbportmgr    536870944
ndbserver1    536870945
ndbserver2    536870946
ndbserver3    536870947
ndbserver4    536870948
ndbserver5    536870949
ndbserver6    536870950
ndbserver7    536870951
ndbserver8    536870952
ndbserver9    536870953
ndbserver10   536870954
ndbserver11   536870955
ndbserver12   536870956
ndbserver13   536870957
ndbserver14   536870958
ndbserver15   536870959
ndbserver16   536870960
ndbserver17   536870961
ndbserver18   536870962
ndbserver19   536870963
ndbserver20   536870964
ndbserver21   536870965
ndbserver22   536870966
ndbserver23   536870967
ndbserver24   536870968
ndbserver25   536870969
ndbserver26   536870970
ndbserver27   536870971
ndbserver28   536870972
ndbserver29   536870973
ndbserver30   536870974
ndbserver31   536870975
ndbserver32   536870976
ndbserver33   536870977
ndbserver34   536870978
ndbserver35   536870979
ndbserver36   536870980
ndbserver37   536870981
ndbserver38   536870982
ndbserver39   536870983
ndbserver40   536870984
ndbserver41   536870985
ndbserver42   536870986
ndbserver43   536870987
ndbserver44   536870988
ndbserver45   536870989
ndbserver46   536870990
ndbserver47   536870991
ndbserver48   536870992
ndbserver49   536870993
ndbserver50   536870994
ndbserver51   536870995
ndbserver52   536870996
ndbserver53   536870997
ndbserver54   536870998
ndbserver55   536870999
ndbserver56   536871000
ndbserver57   536871001
ndbserver58   536871002
ndbserver59   536871003
ndbserver60   536871004
ndbserver61   536871005
ndbserver62   536871006
ndbserver63   536871007
ndbserver64   536871008
ndbserver65   536871009
ndbserver66   536871010
```

```
ndbserver67   536871011
ndbserver68   536871012
ndbserver69   536871013
ndbserver70   536871014
ndbserver71   536871015
ndbserver72   536871016
ndbserver73   536871017
ndbserver74   536871018
ndbserver75   536871019
ndbserver76   536871020
ndbserver77   536871021
ndbserver78   536871022
ndbserver79   536871023
ndbserver80   536871024
ndbserver81   536871025
ndbserver82   536871026
ndbserver83   536871027
ndbserver84   536871028
ndbserver85   536871029
ndbserver86   536871030
ndbserver87   536871031
ndbserver88   536871032
ndbserver89   536871033
ndbserver90   536871034
ndbserver91   536871035
ndbserver92   536871036
ndbserver93   536871037
ndbserver94   536871038
ndbserver95   536871039
ndbserver96   536871040
ndbserver97   536871041
ndbserver98   536871042
ndbserver99   536871043
ndbserver100  536871044
```

Starting the PORTMAP Address Space

If you did not include PORTMAP in the AUTOLOG statement, you can start PORTMAP with the START command. For example:

```
START PORTMAP
```

PORTMAP must be started before NDB can run.

Configuring the z/OS UNIX PORTMAP Address Space

This section describes how to configure the z/OS UNIX PORTMAP address space, which runs the Portmapper function.

Steps to configure z/OS UNIX PORTMAP:

1. Specify PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the PORTMAP cataloged procedure.

Step 1: Configuring PROFILE.TCPIP for UNIX PORTMAP

To ensure that port UDP 111 and TCP 111 is reserved for the z/OS UNIX PORTMAP server, add the name of the member containing the PORTMAP cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  111 UDP OMVS     ; Portmapper Server
  111 TCP OMVS     ; Portmapper Server
```

See the *z/OS Communications Server: IP Configuration Reference* for more information about the PORT statement.

Step 2: Updating the PORTMAP Cataloged Procedure

Update the PORTMAP cataloged procedure to suit your local conditions by copying the sample provided in *hlq*.SEZAINST(OPORTPRC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Change the data set names as required. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details.

Starting the PORTMAP Address Space

There are two ways to start the portmapper as an z/OS UNIX socket application:

- From the z/OS shell
- As a started task

To start the portmapper from the z/OS shell, the user ID must be an authorized superuser. The authorized superuser ID can issue "oportmap &" to start the portmapper. For the authorization procedure, see *z/OS UNIX System Services Planning*.

You can also start PORTMAP as a started task with the START command as follows:

```
START PORTMAP
```

Note: If your system is using the Network File System (NFS) server, see *z/OS Network File System Customization and Operation* for more information.

Configuring the NCS Interface

This section describes how to configure the Network Computing System (NCS).

NCS is the Remote Procedure Call (RPC) implementation of Apollo's Network Computing Architecture (NCA**).

NCS includes:

1. RPC run-time library
2. Location broker
3. Network Interface Definition Language (NIDL) compiler

The RPC runtime library and the location broker provide runtime support. Together these two elements make up the Network Computing Kernel (NCK) which includes all the software necessary to run a distributed application. The NIDL compiler is a tool for developing NCS-based applications.

In order to execute NCS applications in an MVS environment, you must start a local location broker (LLBD). One of the hosts in your TCP/IP network must also start the global location broker (GLBD). Both the LLBD and the NRGLBD maintain information about active NCS server applications.

Understanding the LLBD Server

The LLBD manages the LLB database which stores information for the NCS-based servers running on this host.

Your host must run LLBD if you want to support the location broker forwarding function or allow remote access to the LLB database. The LLBD function must be started on the host before any other NCS-based servers are started and before the NRGLBD is started.

The LLB database is stored in the data set ADM@SRV.LLB@LL.DATABASE. It is not created until an entry is registered with the LLBD.

Understanding the NRGLBD Server

The NRGLBD manages the NCS global location broker (GLB) database. The GLB helps clients locate servers on a network or internet.

There are two versions of the GLB daemon: replicated GLBD and NRGLBD. The replicated GLBD is only available on Apollo, SunOS, and Ultrix systems. For other systems, such as IBM, only the NRGLBD is available. The advantage of replicated GLBD over NRGLBD is that the GLBD can be run at the same time on several network hosts, providing greater availability in the event that one of the hosts running GLBD fails or if there is a partial network failure.

You cannot use the NRGLBD on your system if:
- The replicated version of GLBD can run on some other host in your network
- Another host in your network is already running NRGLBD

The GLB database is stored in a data set ADM@SRV.LLB@LG.DATABASE. This data set is not created until an entry is registered with the NRGLBD.

The record structure for the LLBD and the NRGLBD databases is identical.

Steps to configure NCS:
1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the NRGLBD cataloged procedure in *hlq*.SEZAINST(NRGLBD).
3. Update the LLBD cataloged procedure in *hlq*.SEZAINST(LLBD).

For more information about NCS, see *Network Computing System Reference Manual*.

Step 1: Configuring PROFILE.TCPIP for NCS

If you want LLBD and NRGLBD to start automatically when the TCPIP address space is started, then you should include the names of the members containing the LLBD and NRGLBD cataloged procedures in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

The LLBD must be running before you start NRGLBD. Therefore, you must put LLBD before NRGLBD in the AUTOLOG statement.

```
AUTOLOG
  LLBD
  NRGLBD
ENDAUTOLOG
```

To ensure that port UDP 135 is reserved for the LLBD server, add the name of the member containing the LLBD cataloged procedure to the PORT statement in the *hlq*.PROFILE.TCPIP data set.

```
PORT
  135 UDP LLBD
```

Note: z/OS UNIX DCE also uses port 135 and therefore cannot be used concurrently with NCS.

See the *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Updating the NRGLBD Cataloged Procedure

Update the NRGLBD cataloged procedure by copying the sample provided in *hlq*.SEZAINST(NRGLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions. Refer to the NCS chapter in *z/OS Communications Server: IP Configuration Reference* for more details.

Step 3: Updating the LLBD Cataloged Procedure

Update the LLBD cataloged procedure by copying the sample provided in *hlq*.SEZAINST(LLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions. Refer to the NCS chapter in *z/OS Communications Server: IP Configuration Reference* for more information.

Configuring the Network Database (NDB) System

This section describes how to configure the Network DataBase (NDB) System. NDB provides access to mainframe relational database systems using TCP/IP and the remote procedure call (RPC) protocol. This allows interoperability among a variety of workstation and mainframe users with DB2 and MVS. End users in a VM, MVS, DOS, OS/2, RISC System/6000 AIX, or Sun UNIX Microsystems environment can issue SQL statements interactively or invoke NDB services from within a C application program. NDB services can then be used to pass SQL statements to DB2 and to handle responses from DB2.

The Network DataBase (NDB) System consists of:
- The Network Database (NDB) Port Manager
- The NDB Port Client
- 1–100 NDB Servers
- NDB Clients

Note: The NDB System requires the Portmapper to run. See “Configuring the PORTMAP Address Space” on page 481 for further information.

Steps to Configure the NDB System

1. Update the NDB setup sample job (NDBSETUP).
2. Run the NDB setup sample job.
3. Update and install the DB2 sample connection exit routine (DSN3SATH), if necessary.
4. Update the PORTS cataloged procedure (PORTSPRC).
5. Update the PORTC cataloged procedure (PORTCPRC).
6. Create the NDB clients.

Note: Repeat steps 1 and 2, and step 3 if necessary, to configure each DB2 subsystem that is to be accessed by NDB servers.

Step 1: Updating the NDB Setup Sample Job

Update the NDBSETUP sample job by copying the sample in *hlq*.SEZAINST(NDBSETUP) to a partitioned data set (PDS) or sequential data set

and modifying it to suit your local installation. Change the DD statements as required. A copy of NDBSETUP can also be found *z/OS Communications Server: IP Configuration Reference*.

To restrict the use of NDB to selected users, modify the GRANT statement in NDBSETUP job replacing PUBLIC with a list of the specified TSO user IDs to whom you want to grant access. See the *DB2 SQL Reference* for correct statement syntax.

THE NDBSETUP job binds the DBRM DBUTIL2 into the DB2 subsystem specified. In previous releases, the DBRM was bound using the plan name DBUTIL2. For V3R2, the DBRM is bound using the plan name EZAND320. By using different plan names for each release, you can have different levels of TCP/IP similutaneously accessing the same DB2 subsystems.

Note: NDBSETUP binds the DBRM DBUTIL2 with the isolation level of cursor stability: ISOLATION(CS) on the BIND command. If a higher degree of unit of work integrity is needed than is provided by an isolation level of cursor stability, this can be changed to an isolation level of repeatable read: ISOLATION(RR) on the BIND command. Refer to the *DB2 Application Programming and SQL Guide* for information regarding isolation level settings and their effects. Check with your database administrator regarding isolation level policies at your site.

Step 2: Running the NDB Setup Job

Run NDBSETUP to bind the DBRM DBUTIL2 to the DB2 subsystem specified and grant the execute (run) privileges to public. Verify that it ran successfully before proceeding with this configuration.

Notes:

1. The DB2 subsystem must be Version 2 Release 3 or higher. This requirement applies to all TCP/IP servers that use DB2.
2. The DB2 subsystem must be up and running before you submit the NDBSETUP job.
3. The NDBSETUP job must be executed from a user ID with the authority to bind plans for the specified DB2 subsystem.

Step 3: Updating and Installing the DB2 Sample Connection Exit Routine

Note: If you will be running only one NDB server in a single address space, you may skip this step.

Because the address space running a PORTC cataloged procedure is capable of running between 1 to 20 NDB servers, changes must be made in the DB2 sample connection exit routine, DSN3SATH ASSEMBLE. These changes tell DB2 where to look for the host userid to be used for this session (the host userid was supplied by the user at NDB client invocation).

After making the necessary changes in DSN3SATH ASSEMBLE, this routine can be assembled and link edited using the DB2 supplied job DSNTIJEX in the DSNSAMP library. See DSN3SATH ASSEMBLE Modifications for NDB for the updated assembler code. For more infomation, see the *DB2 Administration Guide* DSNTIJEX will place a copy of the executable in the DSNEXIT library. There is one DSNEXIT library for each DB2 subsystem and it must be updated with the recommended

modifications for each DB2 subsystem that will be accessed via NDB. Run DSNTIJEX once (updating the hlq for the DSNEXIT library) for each DB2 subsystem that will be accessed via NDB.

DSN3SATH ASSEMBLE Modifications for NDB

```
⋮
**********SECTION 1:  DETERMINE THE PRIMARY AUTHORIZATION ID  *********
*                                                                     *
*   IF THE INPUT AUTHID IS NULL OR BLANKS, CHANGE IT TO THE AUTHID    *
*   IN EITHER THE JCT OR THE FIELD POINTED TO BY ASCBJBNS.            *
*                                                                     *
*   THE CODE IN THIS SECTION IS AN ASSEMBLER LANGUAGE VERSION OF      *
*   THE DEFAULT IDENTIFY AUTHORIZATION EXIT.  IT IS EXECUTED ONLY     *
*   IF THE FIELD ASXBUSER IS NULL UPON RETURN FROM THE RACROUTE       *
*   SERVICE.  FOR EXAMPLE, IT DETERMINES THE PRIMARY AUTH ID FOR      *
*   ENVIRONMENTS WITH NO SECURITY SYSTEM INSTALLED AND ACTIVE.        *
*                                                                     *
***********************************************************************
         SPACE
* MOVED CHECK ON AIDLPRIM THAT WAS HERE TO PAST CHECK FOR TSO FOREGRND
         L     R7,ASCBCSCB         GET CSCB ADDRESS
         CLI   CHTRKID-CHAIN(R7),CHTSID IS IT TSO FOREGROUND ADDR SPACE
         BNE   SATH010             BRANCH IF NOT
* HERE IS NEW LOCATION OF CHECK ON AIDLPRIM
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BH    SATH020             SKIP IF A PRIMARY AUTH ID EXISTS
* END OF MOVED CODE - CHECK ON AIDLPRIM
         L     R7,ASCBJBNS         GET ADDRESS OF LOGON ID
         MVC   AIDLPRIM,0(R7)      MAKE IT THE PRIMARY AUTH ID
         B     SATH019             TO END OF THIS ROUTINE
SATH010  DS    0H                  NOT TSO, BUT BATCH OR STC SPACE
         L     R6,PSATOLD-PSA      CURRENT TCB ADDRESS
* START OF CODE ADDED TO SUPPORT TCP/IP NDB MULTI DB2 SUBSYSTEM FUNCT
         USING TCB,R6
         L     R7,TCBSENV          CURRENT ACEE ADDRESS
         USING ACEE,R7
         CLC   ACEEACEE,EYEACEE    IS THIS AN ACEE?
         BNE   SATH015             NO, GO USE JOB USER ID
         L     R14,TCBJSCB         ADDRESS OF JSCB
         USING IEZJSCB,R14
         CLC   JSCBPGMN,=CL8'PORTCLNT' IS IT NDB SERVER AS ?
         BNE   SATH015             NO, GO USE JOB USER ID
         MVC   AIDLPRIM,ACEEUSRI   MOVE USER ID INTO AIDL PRIMARY
         MVC   AIDLACEE,TCBSENV    MOVE ACEE POINTER INTO AIDL
         B     SATH020
         DROP  R6,R7,R14
SATH015  DS    0H
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BH    SATH020             SKIP IF A PRIMARY AUTH ID EXISTS
* END OF CODE ADDED TO SUPPORT TCP/IP NDB MULTI DB2 SUBSYSTEM FUNCTION
         L     R7,TCBJSCB-TCB(,R6) CURRENT JSCB ADDRESS
         L     R5,JSCBJCT-IEZJSCB(,R7) CURRENT JCT ADDRESS
         LA    R5,X'10'(,R5)       ADJUST FOR CORRECT DSECT MAPPING
         MVC   AIDLPRIM(7),JCTUSER-INJMJCT(R5)  COPY JOB USER ID
         MVI   AIDLPRIM+7,BLANK    ASSURE BLANK PADDING
SATH019  DS    0H                  END OF ROUTINE
         EJECT
⋮
```

Step 4: Updating the PORTS Cataloged Procedure

The PORTS procedure starts the NDB Port Manager. Update the PORTS cataloged procedure by copying the sample in *hlq*.SEZAINST(PORTSPRC) to your system or

recognized PROCLIB and modifying it to suit your local installation. Change the DD statements, as required. Refer to *z/OS Communications Server: IP Configuration Reference* for the PORTS procedure.

Note: You must start PORTS before you start PORTC.

Step 5: Updating the PORTC Cataloged Procedure

The PORTC procedure starts the NDB Port Client and the NDB Servers. Update the PORTC cataloged procedure by copying the sample in *hlq*.SEZAINST(PORTCPRC) to your system or recognized PROCLIB and modifying it to suit your local installation. Refer to the *z/OS Communications Server: IP Configuration Reference* for a copy of PORTC.

Running Multiple PORTC Procedures

Currently the NDB Port Manager is able to manage up to 100 NDB Servers. Each PORTC procedure is able to start up to 20 NDB servers. The number of NDB Servers that can be started in a single address space depends on a number of factors, such as how large a region size can be specified and the size of the catalog work area. These factors may limit the number of NDB Servers able to start in a single address space to less than the maximum of 20.

In order to reach the desired number of NDB Servers, you can run multiple PORTC procedures. When starting a PORTC procedure, one of the parameters specified is DB2SSID. See *z/OS Communications Server: IP Configuration Reference* for more information on the DBSSID parameter. This parameter indicates which DB2 subsystem all NDB servers running in that address space will access. Each started PORTC procedure may contain a different value for the DBSSID parameter; that is, when starting multiple PORTC procedures, each procedure may point to the same or different DB2 subsystems. To do this:

1. Copy your customized PORTC cataloged procedure to another data set or PDS member.
2. Change the name in the first statement (PROC statement). For example if your original PORTC procedure starts with //PORTC PROC, use //PORTC1 PROC and //PORTC2 PROC for the other procedures.
3. Ensure that:
   - HOMEID parameter settings are the same for all PORTC procedures
   - When the NUMSRV parameter settings for all the PORTC procedures are added together, the total does not exceed 100
   - The NUMSRV value for any given PORTC does not exceed 20.
4. Start the new PORTC procedures.

Step 6: Creating the NDB Clients

z/OS CS provides NDB sample client code that can be compiled and run in a variety of environments. NDB clients can be created on VM, MVS, or on DOS, OS/2, AIX on RS/6000, or SUN UNIX workstations. You can find the NDB client source code in the *hlq*.SEZAINST data set, unless otherwise noted. They are written in the C language.

The process to create a client is similar in each case. Instructions for creating a client in each of the supported environments are specified. The MVS client code shipped with TCP/IP is ready to use without modification but you can modify it, if necessary, to suit your environment.

Note: To build an NDB client, you need access to the C runtime libraries and the TCP/IP RPC libraries. The TCP/IP products for the client platforms must be installed on those platforms before you build the NDB clients. In addition, for the DOS and OS/2 platforms, you must also install the TCP/IP Programmer's Toolkit.

Creating an NDB Client in the AIX Environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Issue the command:

```
cc -o ndbclnt ndbmain.c ndbc.c ndbclt.c ndbout.c ndbpclt.c ndbx.c
```

Notes:

1. This version of the NDB sample client code was produced and tested on AIX Version 3 Release 2 for RISC System/6000.
2. The file NDBRPCFC (ndbrpcf.c) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new copy before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See *TCP/IP AIX Programmer's Reference* for more information on the RPCGEN process.

Creating an NDB Client in the SUN UNIX Environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Issue the command:

```
cc -o ndbclnt -DNOPROTO ndbmain.c ndbc.c ndbclt.c ndbout.c ndbx.c
   ndbpclt.c
```

Notes:

1. This version of the NDB sample client code was produced and tested on SUN OS Version 4 Release 1 Modification 1.
2. The file NDBRPCFC (ndbrpcf.c) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new copy before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See the *SUN Network Programming* manual for more information.

Creating an NDB Client in the OS/2 Environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBRPCFC	ndbrpcf.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBCLTH	ndbclt.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Bring the following OS/2 files down to your workstation:

Program	Rename to
NDBCLDEF	ndbclnt.def
NDBOS2M	ndbos2.mak

NDBOS2M has been stored as a binary file on MVS and, as such, cannot be read on the MVS host. When using FTP to move this file from MVS to OS/2, make sure the transfer type used is BINARY. The OS/2 makefile (NDBOS2M) is in the SEZAXLD2 data set.

4. Issue the command:

```
nmake -f ndbos2.mak
```

Notes:

1. This version of the NDB sample client code was produced and tested on OS/2 Warp Version 3.0, TCP/IP for OS/2 Version 3.0, including the TCP/IP for OS/2 Programmer's Toolkit, and IBM C Set ++ Version 2.0.
2. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the z/OS TCP/IP library, do the following to generate a new one before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See *TCP/IP for OS/2 Programmer's Reference* for more information.

Creating an NDB Client in the DOS Environment

1. Bring the following C source programs down to your workstation

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBRPCFC	ndbrpcf.c
NDBXC	ndbx.c

   When using FTP to move these files from MVS to DOS, make sure the transfer type used is ASCII.

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

   When using FTP to move these files from MVS to DOS, make sure the transfer type used is ASCII.

3. Bring the following DOS makefile down to your workstation:

Program	Rename to
NDBDOSM	ndbdos.mak

NDBDOSM has been stored as an binary file on MVS and, as such, cannot be read on the MVS host. When using FTP to move this file from MVS to DOS, make sure the transfer type used is BINARY. The DOS makefile (NDBDOSM) is in the SEZAXLD2 data set.

4. Issue the command:

```
nmake -f ndbdos.mak
```

Notes:

1. This version of the NDB sample client code was produced and tested on IBM DOS Version 6.0, TCP/IP for DOS Version 2.1.1.4, including the TCP/IP for DOS Programmer's Toolkit [3], Microsoft Windows Version 3.1 and Microsoft C/C++ Version 7.0.

   The TCP/IP for DOS Programmer's Toolkit Version 2.1.1.4 contains a fix to the RPC library required to run the NDB DOS client.

2. TCP/IP for DOS does not support RPC server functions because it does not have a PORTMAPPER. Therefore, if the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the z/OS TCP/IP library, do the following to generate a new one on a workstation or mainframe that does support RPC server functions before compiling the NDB sample client code.

   a. Bring the following X (RPC input) file down to your workstation or mainframe:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

   ```
   rpcgen -h -o ndb.h ndb.x
   ```

   c. Transfer the resulting NDB.H file to your DOS workstation

See *TCP/IP DOS Programmer's Reference* , and the Programmer's Reference manuals for the workstation or the mainframe being used for RPCGEN for more information.

Creating an NDB Client in the VM Environment

1. Bring the following C source programs to your VM user ID:

Program	Rename to
NDBCC	NDBC C
NDBCLTC	NDBCLT C
NDBMAINC	NDBMAIN C
NDBOUTC	NDBOUT C
NDBPCLTC	NDBPCLT C
NDBXC	NDBX C

2. Bring the following H (header) files to your VM user ID:

Program	Rename to
NDBCLTH	NDBCLT H
NDBGLOBH	NDBGLOB H
NDBH	NDB H
NDBOUTH	NDBOUT H
NDBPCLTH	NDBPCLT H
NDBRPCFH	NDBRPCF H

3. Bring the following Assembler file to your VM user ID:

---

3. The TCP/IP for DOS Programmer's Toolkit Version 2.1.1.4 contains a fix to the RPC library that is needed for the NDB DOS client to run.

Program	Rename to
NDBVMPWA	NDBVMPW ASSEMBLE

4. Bring the following VM REXX EXEC to your VM user ID:

Program	Rename to
NDBCLBD	NDBCLBD EXEC

5. Execute the EXEC by entering:

```
ndbclbd
```

Notes:

1. This version of the NDB Sample Client code was produced and tested on VM/ESA® Version 1 Release 2 using CMS Level 9, running in XA mode and in ESA mode.
2. The file NDBRPCFC (NDBRPCF C) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (NDB H) file, an output of the RPCGEN process, should be erased or corrupted and cannot be retrieved from the MVS TCP/IP library, you will need to generate a new one before you can successfully compile the NDB sample client code. To do this:
   a. Bring the following X (RPC input) file to your VM user ID:

Program	Rename to
NDBXX	NDB X

   b. Issue the command:

```
rpcgen -h -o ndb h ndb x
```

See *TCP/IP for VM Programmer's Reference* for further information.

Creating an NDB Client in the MVS Environment

1. Copy the following C source programs to a PDS:

Program	Rename to
NDBCC	NDBC
NDBCLTC	NDBCLT
NDBMAINC	NDBMAIN
NDBOUTC	NDBOUT
NDBPCLTC	NDBPCLT
NDBXC	NDBX

2. Copy the following H (header) files to a PDS:

Program	Rename to
NDBCLTH	NDBCLT
NDBGLOBH	NDBGLOB
NDBH	NDB
NDBOUTH	NDBOUT
NDBPCLTH	NDBPCLT
NDBRPCFH	NDBRPCF

3. Copy the following sample JCL to a data set or PDS and customize it following the instructions found in the sample:
   NDBCLMVS
4. Execute the JCL.

Notes:

1. This version of the NDB Sample Client code was produced and tested on MVS/ESA Version 4 Release 2 Modification 2.

2. The file NDBRPCFC (NDBRPCF C) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (NDB H) file, an output of the RPCGEN process, should be erased or corrupted and cannot be retrieved from the MVS TCP/IP library, you will need to generate a new one before you can successfully build the NDB sample client. To do this:
   a. Copy the following X (RPC input) file to a data set:

Program	**Rename to**
NDBXX	NDB.X

   b. Issue the command:

      ```
      rpcgen -h -o ndb.h ndb.x
      ```

See the *z/OS Communications Server: IP Programmer's Reference*for further information.

Starting NDB

Follow these steps to start NDB:

1. Ensure that the Portmapper is up and running. For more information, see "Configuring the PORTMAP Address Space" on page 481.
2. Run the PORTSPRC procedure to start the NDB Port Manager.
3. After PORTSPRC has started, run the PORTCPRC procedure to start the NDB Port Client and NDB Servers. Specify the start parameters as required.
4. Ensure that the required DB2 subsystem is running.
5. Invoke the NDB client.

Chapter 13. Configuring the Kerberos Server

Before You Configure...

Read “Understanding Search Orders of Configuration Information” on page 6. It covers important information about data set naming and search sequences.

This chapter describes how to configure and verify the Kerberos Authentication System for TCP/IP.

The Kerberos system in TCP/IP consists of the following:
- Kerberos database
- Kerberos authentication server
- Remote database administration server
- Remote database administration client
- Local database administrator utilities
- Service programs
- Client programs

Configuration Process

Steps to configure Kerberos:
1. Add PORT statements to the *hlq*.PROFILE.TCPIP data set.
2. Update the authentication server cataloged procedure.
3. Update the remote database administration cataloged procedure.
4. Define the Kerberos services in *hlq*.ETC.SERVICES.
5. Create and update the Kerberos system data sets.
6. Authorize Kerberos servers.
7. Build the Kerberos database.
8. Store the Master Key.
9. Start the Kerberos servers.

A list of background information about Kerberos can be found in the bibliography. See *z/OS Communications Server: IP Programmer's Reference* for more information on the application programming interface. See *z/OS Communications Server: IP User's Guide* for additional commands anddescriptions of the Kerberos functions.

Step 1: Add PORT Statements to the hlq.PROFILE.TCPIP Data Set

To ensure that ports TCP 750, UDP 750, TCP 751, and UDP 751 be reserved for Kerberos, add the names of the members containing the Kerberos cataloged procedures to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  750 TCP MVSKERB
  750 UDP MVSKERB
  751 TCP ADM@SERV
  751 UDP ADM@SERV
```

See *z/OS Communications Server: IP Configuration Reference* for more information about this statement.

Kerberos should not be included in the AUTOLOG list as a procedure to be started automatically when TCP/IP is invoked. Kerberos does not have a listening connection on its reserved ports and, if placed in the AUTOLOG list, would be subject to being periodically cancelled and restarted by TCP/IP.

Step 2: Update the Authentication Server Cataloged Procedure

Update the MVSKERB cataloged procedure by copying the sample in *hlq*.SEZAINST(MVSKERB) to your system or recognized PROCLIB. Specify MVSKERB parameters and change the DD statements, as required, to suit your local configuration.

Step 3: Update the Remote DBA Server Cataloged Procedure

Update the remote database administration (DBA) cataloged procedure by copying the sample in *hlq*.SEZAINST(ADM@SERV) to your system or recognized PROCLIB and modifying it to suit your local configuration. Specify ADM@SERV parameters and change the DD statements, as required.

Step 4: Define the Kerberos Services in hlq.ETC.SERVICES

Before you set up the Kerberos system, the Kerberos services must be defined and their port numbers established. Add the following lines to the *hlq*.ETC.SERVICES data set:

```
kerberos             750/tcp
kerberos             750/udp
kerberos_master      751/tcp
kerberos_master      751/udp
```

The entries in *hlq*.ETC.SERVICES are case and column sensitive. They must be in lowercase and begin in column 1.

Step 5: Create and Update the Kerberos System Data Sets

This step describes the data sets that you must create and update for the Kerberos system.

Kerberos Configuration Data Set (KRBCONF)

You must create either a *user_id*.KRB.CONF data set (where *user_id* is the member or proc name of the Kerberos authentication server) or an ADM@SRV.KRB.CONF data set. This data set identifies the hosts that are running the Kerberos authentication server. The *user_id*.KRB.CONF data set must be accessible by client applications.

The format of the data set is:

```
realm
realm host_name
realm host_name admin server
```

The contents of the *user_id*.KRB.CONF data set are case-sensitive.

Some examples of the data set format are:

```
univ.educ.chem
univ.other.dept joanpc
univ.educ.math chrispc admin server
```

The first line defines the local *realm* to which that host belongs. Each of the following lines specify the *realm* and *host_name* where the Kerberos server is running.

In our example, the third line lists `admin server` to indicate that the host provides an administration database server. The *host_name* that you specify must also be defined in your Domain Name Server or in the *hlq*.HOSTS.LOCAL data set.

The following sample data set can be copied from *hlq*.SEZAINST(KRBCONF).

```
raleigh.ibm.com
raleigh.ibm.com mvs2
raleigh.ibm.com mvs2 admin server
```

Note: You cannot use dotted decimal IP addresses, such as 9.67.60.11, in the KRB.CONF data set.

Kerberos Remote DBA Authorization Data Sets

To authorize the remote database administrator to add, get, and modify database entries, you must create the remote administrator authorization data sets. The data set names are:

- ADM@ACL.ADD
- ADM@ACL.GET
- ADM@ACL.MOD

You can copy these data sets from the corresponding samples:

- *hlq*.SEZAINST(ADMADD) for ADM@ACL.ADD
- *hlq*.SEZAINST(ADMGET) for ADM@ACL.GET
- *hlq*.SEZAINST(ADMMOD) for ADM@ACL.MOD

These data sets should be accessed by the remote database administration server. Your Kerberos name must be in these data sets if you want to use the KADMIN command to add, get, or modify Kerberos databases remotely.

The format of these data sets is:

administrator's_principal_name.admin@*realm*

realm is usually the domain name. The contents of these data sets are case-sensitive.

These data sets can contain multiple entries in the same format. Some examples of the data set format are:

```
krbadd.admin@univ.educ.chem
krbget.admin@univ.educ.bio
krbmod.admin@univ.educ.math
```

Step 6: Authorize Kerberos Servers

Using IBM's RACF program or your own security program, you need to authorize the two servers for access to the Kerberos database:

- The authentication server address space must have ***read*** access to the Kerberos database.
- The remote database administration server address space must have ***read/write*** access to the Kerberos database.

Step 7: Build the Kerberos Database

A special user ID named ADM@SRV must be defined in your system.

You must enter the Kerberos commands from the ADM@SRV user ID to create and use the Kerberos database before you can start the Kerberos administration server and Kerberos authentication server.

Follow these steps to create the Kerberos database:

1. Issue the KDB@INIT command to create and initialize the Kerberos database data sets. The system prompts you for the local realm.

2. At the realm prompt, enter the name of the realm where the Kerberos database resides. This is the local realm specified in the first line of the *user_id*.KRB.CONF data set. The default is YOUR_KRB.REALM. The system prompts you for the master key.
3. At the prompt, enter the master key.

   You need the master key to manage the Kerberos database. If the Kerberos database data set already exists, the system indicates that the data set already exists. Use KDB@DEST to destroy the existing database data set. See *z/OS Communications Server: IP Configuration Reference* for more information.

KDB@INIT creates three database data sets:
- ADM@SRV.PRINCPL.DAT
- ADM@SRV.PRINCPL.IDX
- ADM@SRV.PRINCPL.OK

See the *z/OS Communications Server: IP Configuration Reference* for more information.

Step 8: Store the Master Key

The following steps describe how to store the master key.

1. On the *master_host* ADM@SRV user ID and from TSO, issue the KSTASH command.

   The system prompts you for the master key.
2. Enter `krbpass` at the password prompt.

   See *z/OS Communications Server: IP Configuration Reference* for more information.

Step 9: Start the Kerberos Servers

This section describes how to start the following Kerberos servers:
- Authentication server
- Remote database administration server

Start the Kerberos Authentication Server

The Kerberos authentication server provides a way for authenticated users to prove their identity to other servers across a network. The authentication server reads the Kerberos database to verify that the client making the request is the client named in the request. For more information about the authentication server, which includes the ticket-granting server, see *z/OS Communications Server: IP Programmer's Reference*.

Before you can use the Kerberos system, you must start the Kerberos authentication server. See "Step 2: Update the Authentication Server Cataloged Procedure" on page 498 for instructions on setting up the cataloged procedure for this server.

Use the member name of the authentication server procedure as the *procname* in the START command.

The following options are available when starting the Kerberos authentication server:

-r Indicates the realm.

-l Indicates the file name. The file name must be a fully qualified data set name.

After the authentication server is started, you can use the Kerberos system. See the *z/OS Communications Server: IP User's Guide* for a description of the Kerberos user commands.

Maintaining the Log: When the authentication server starts, it creates a log data set called ADM@SRV.KERBEROS.LOG. All transactions to the Kerberos authentication server are recorded in this data set. Because Kerberos appends continuously, you should periodically monitor the size of the ADM@SRV.KERBEROS.LOG data set.

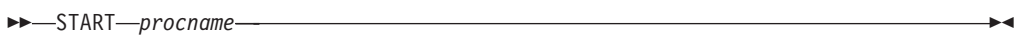
Stopping the Authentication Server: The authentication server can be stopped using the MVS STOP command. The authentication server will be terminated automatically when the TCP/IP address space is terminated.

Start the Kerberos Remote DBA Server

The Kerberos remote database administration server allows you to modify the Kerberos database remotely. Kerberos database data sets are accessed by the remote database administration server in write mode.

Before you can use the KADMIN command, you must start the remote database administration server. See "Step 3: Update the Remote DBA Server Cataloged Procedure" on page 498 for instructions on setting up the cataloged procedure for this server.

Use the member name of the remote database administration server procedure as the *procname* in the START command.



After the remote database administration server is started, you can use the KADMIN command on a remote host to add, retrieve, or modify the Kerberos database. See *z/OS Communications Server: IP Configuration Reference* for more information about the KADMIN command.

Maintaining the Log: When the remote database administration server starts, it creates a log data set called ADM@SRV.ADM@SRV.LOG. All transactions to this server are recorded in this data set. Because Kerberos appends continuously, you should periodically monitor the size of the ADM@SRV.ADM@SRV.LOG data set.

Stopping the Remote Database Administration Server: The remote database authentication server can be stopped using the MVS STOP command. The remote database authentication server will be terminated automatically when the TCP/IP address space is terminated.

Verifying the Kerberos Configuration

TCP/IP for MVS provides a sample application client program and a sample application server program that can be used to verify your Kerberos installation and configuration.

This section shows an example of how to set up, run, and verify a Kerberos system.

Steps to verify Kerberos:
1. Set up the environment.
2. Register the sample service and the user.
3. Generate the key data set for the sample service.
4. Transfer the service key data set to the server.
5. Start the sample server.
6. Get the initial ticket.
7. Run the sample client program.

In this example, the hosts are assigned the following names:

Host Name	Definition
service_host	Specifies the name of the host on which the SAMPLE@S server is running.
client_host	Specifies the name of the host on which the SAMPLE@C client program is running.
master_host	Specifies the name of the host on which the Kerberos servers and the Kerberos database are running.

Note: If you run the SAMPLE@S and SAMPLE@C programs on the host where the Kerberos servers are running, you will need three IDs on the same host for verification. Verify that each host is set up correctly.

Step 1: Set Up the Environment

Follow steps 1 through 8 in "Configuration Process" on page 497.

Start the Kerberos authentication server as described in "Start the Kerberos Authentication Server" on page 500.

Note: If you fail to start the authentication server, the error messages can be seen in the output from the console.

Step 2: Register the Sample Service and the User

The following steps describe how to register the service and user.

1. On the *master_host*.ADM@SRV user ID, and from TSO, issue the KDB@EDIT command.

   The system prompts you for the Kerberos master password.
2. Enter `krbpass` at the password prompt.
3. If the master password is valid, KDB@EDIT allows you to register a service.

   The system prompts you for the *principal_name*.
4. Enter `sample` at the principal name prompt. `sample` is the service name.

   The system prompts you for the instance.
5. Enter *watson* at the instance prompt.

The message `<not found>` is displayed and the `Create 'y'` prompt is displayed.

6. Enter a null character at the `Create 'y'` prompt to use the default value.
   The system prompts you for the password.
7. Enter `sam` at the password prompt.
   For verification purposes, the system prompts you again for the password.
8. Reenter `sam` at the password prompt for verification.
9. Enter a null character to use the default values for the remaining prompts.
   The following message is displayed if service, `sample`, is registered:
   ```
   Edit O.K.
   ```
   The system prompts you for the *principal_name*.
10. Enter `user1` at the principal name prompt.
    The system prompts you for the instance.
11. Enter a null character for a null instance at the instance prompt.
    The message `<not found>` is displayed and the `Create 'y'` prompt is displayed.
12. Enter a null character at the `Create 'y'` prompt to use the default value.
    The system prompts you for the password.
13. Enter `use` at the password prompt.
    For verification purposes, the system prompts you again for the password.
14. Reenter `use` at the new password prompt for verification.
15. Enter a null character to use the default values for the remaining prompts.
    The following message is displayed if user, `user1` is registered:
    ```
    Edit O.K.
    ```
    The system prompts you for the *principal_name*.
16. Enter a null character at the principal name prompt to exit the registration process.

See *z/OS Communications Server: IP Configuration Reference* for more information about the KDB@EDIT command.

Note: *Service* refers to the host name on which SAMPLE@S is to be run.

Step 3: Generate the Key Data Set for the Sample Service

The following steps describe how to generate the key data set for the sample service.

1. On the *master_host.*ADM@SRV user ID, enter `EXT@SRVT` *watson*.
   The system prompts you for a password.
2. Enter `krbpass` at the password prompt.
3. A key data set ADM@SRV.WATSON.SRVTAB is created that contains the service keys provided by *service_host*.
   See the *z/OS Communications Server: IP Configuration Reference* for more information about the EXT@SRVT command.

Step 4: Transfer the Service Key Data Set to the Server

The following steps describe how to transfer the key data set to the *server_host*.

1. Use the FTP program to transfer the ADM@SRV.WATSON.SRVTAB key data set from the *master_host* to the *service_host* as a binary transfer (EBCDIC/mode b).
2. Rename the key data set.
   - If the host providing the services is MVS, you must rename the key data set *service_id*.ETC.SRVTAB, where *service_id* is the user ID that will be starting the server.
   - If the host providing the services is VM, you must rename the key file ETC SRVTAB.
   - If the host providing the services is OS/2, DOS, UNIX, or AIX, you must rename the key file SRVTAB in the ETC directory.

Step 5: Start the Sample Server

The following describes how to start the sample server, SAMPLE@S.

1. In the server's *hlq*.ETC.SERVICES data set, verify that the following entries exist:

   ```
   sample      906/tcp
   sample      906/udp
   ```

   where 906 is the port number that the server is using.
2. On the *service_host* or from a TSO user ID at the TSO READY prompt, invoke SAMPLE@S, to start the sample Kerberos service application.

Note: If SAMPLE@S is running in the foreground, it will run until you press the attention interupt key (on some keyboards, the attention interrupt key is ATTN). If SAMPLE@S is running in the background, it will run until you issue CANCEL.

Step 6: Get the Initial Ticket

The following steps describe how you get the initial ticket.

1. On the *client_host* or TSO user ID, issue the KINIT command to get the initial ticket.

   The system prompts you for the Kerberos name.
2. Enter `user1` at the Kerberos name prompt.

   The system prompts you for a password.
3. Enter `use` at the password prompt.

   If the KINIT command is successful, the *user_id*.TMP.TKT0 ticket data set is generated on the client host. If the KINIT command is not successful an error message is issued.

See the *z/OS Communications Server: IP User's Guide* for the format and description of the parameters of the KINIT command.

Step 7: Run the Sample Client Program

To start the sample Kerberos client program, enter the following on the *client_host*:

```
SAMPLE@C service_host 100
```

If all parts of your Kerberos system (environment, Kerberos database, Kerberos authentication server, and client and service applications) are set up correctly, a message is displayed as a return from the SAMPLE@S program. The following is an example of the message that might be displayed:

```
The server says:
You are user1.@UNIV.DEPT.BIO (local name user1),
at address 9.67.43.74, version VERSION X, cksum 100.
```

Setting Up a Service or Client Application

You can develop your Kerberos services and clients' applications by referencing SAMPLE@S and SAMPLE@C programs, which are provided in the *hlq*.SEZAINST data set.

You must assign port numbers for the service in the *hlq*.ETC.SERVICES data set. We recommend that you use the port numbers that are defined in the sample *hlq*.SEZAINST(SERVICES).

Setting Up a Service Application

Use the following steps to set up a service application:

1. The database administrator uses KDB@EDIT to register the service name with the Kerberos database. The service name is used as the principal and the host name where the service is running as the instance. For example, the FTP server on the host `chrispc` would be registered as:

   `ftp.chrispc`

   You should also provide a password for the service you register. This password is converted to the key for the service.

   After you register all the services provided by the same host, enter the command:

   `EXT@SRVT` *`instance`*

   For example, `EXT@SRVT chrispc` generates the ADM@SRV.CHRISPC.SRVTAB data set. The server's keys will be stored in this data set.
2. Transfer the key data set to the host providing the services (chrispc in our example).
   - You must rename the key data set *service_id*.ETC.SRVTAB, if the host providing the services is MVS.
   - You must rename the key data set ETC SRVTAB *fm*, where *fm* is the applicable file mode, if the host providing the services is VM.
   - You must rename the key data set SRVTAB in ETC directory, if the host providing the services is OS/2, DOS, UNIX, or AIX.
3. You can now start the service on the host providing the services (for example, `START chrispc`).
4. The service name must be defined in the *hlq*.ETC.SERVICES data set.

Setting Up a Client Application

The following steps describe how to set up a client application.

1. The database administration should register the client with the Kerberos database locally using KDB@EDIT or remotely using the KADMIN function.
2. Verify that the *user_id*.KRB.CONF data set at the client address space contains a valid entry. For more information see "Step 5: Create and Update the Kerberos System Data Sets" on page 498.

3. Issue the KINIT command to get an initial ticket. The ticket is saved in the *user_id*.TMP.TKT0 data set.
4. You can now start your Kerberos client application.
5. Use the KLIST command to see the ticket in the client's ticket data set. The client *user_id*.TMP.TKT0 data set contains tickets used by client applications for different servers.

   You can use the KDESTROY command to delete the ticket data set.

   See *z/OS Communications Server: IP User's Guide* for the format and description of the parameters of the KINIT, KLIST, and KDESTROY commands.

Administrative Commands for the Kerberos Database

Table 19 shows the Kerberos commands you can use to create and administer a Kerberos database. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Table 19. Summary of Kerberos Database Commands

Statement	Description
EXT@SRVT	Generates key data sets for specified instances from ADM@SRV user ID
KADMIN	Adds, retrieves, or modifies the Kerberos database remotely from any ID
KDB@DEST	Erases Kerberos database data sets from ADM@SRV user ID
KDB@EDIT	Registers users to the Kerberos database from ADM@SRV user ID
KDB@INIT	Builds and formats the Kerberos database from ADM@SRV user ID
KDB@UTIL	Loads or dumps the Kerberos database from ADM@SRV user ID
KSTASH	Stores the Master Key

Chapter 14. Mail Servers

Configuring the SMTP Server

Before You Configure...

Read “Understanding Search Orders of Configuration Information” on page 6. It covers important information about data set naming and search sequences.

Note: Before configuring the SMTP server, it is assumed that the necessary SYS1.PARMLIB change have been made. Consult the Program Directory for current information about the storage estimates for this version. The Program Directory also contains information about customization of certain SYS1.PARMLIB members, which must be completed before the initial program load (IPL) for the MVS image.

This chapter describes how to configure the Simple Mail Transfer Protocol (SMTP) server. There is also a section about operating the SMTP server.

The (SMTP or LPD) server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E will be issued and the server will terminate.

Configuration Process

Steps to configure SMTP:

1. Specify AUTOLOG and PORT statements in the *hlq*.PROFILE.TCPIP data set.
2. Update the SMTP cataloged procedure *hlq*.SEZAINST(SMTPPROC).
3. Customize the SMTPNOTE CLIST and modify PARMLIB members.
4. Customize the SMTP mail headers (optional).
5. Set up a TCP-to-NJE mail gateway (optional).
6. Specify configuration statements in the SMTP configuration data set.
7. Create an SMTP security table (optional).
8. Enable SMTP domain name resolution.
9. Enable sending messages to SMTP users and users on an IP Network.

Step 1: Specify AUTOLOG and PORT Statements in hlq.PROFILE.TCPIP

If you want the SMTP server to start automatically when the TCPIP address space is started, include the name of the member containing the SMTP cataloged procedure in the AUTOLOG statement of the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  SMTP
ENDAUTOLOG
```

To ensure that port TCP 25 is reserved for SMTP, verify that the name of the member containing the SMTP cataloged procedure has been added to the PORT statement in *hlq*.PROFILE.TCPIP.

```
PORT
  25 TCP SMTP
```

For more information on the AUTOLOG and PORT statements, see *z/OS Communications Server: IP Configuration Reference*.

Step 2: Update the SMTP Cataloged Procedure

Update the SMTP cataloged procedure by copying the sample in *hlq*.SEZAINST(SMTPPROC) to your system or recognized PROCLIB. Specify SMTP parameters, and change the data set as required to suit your local configuration. Refer to *z/OS Communications Server: IP Configuration Reference* for more detailed information about the procedure.

Note: SMTP does not support HFS files.

Step 3: Customize the System CLIST and Modify PARMLIB Data Sets

You must copy and customize the SMTPNOTE CLIST on every system where users will be able to send mail with the SMTPNOTE command. This includes TCP/IP nodes and each NJE node that sends mail through SMTP on a remote gateway node. SMTPNOTE uses the TSO transmit (XMIT) command to interface with SMTP.

Copy the SMTPNOTE member of the *hlq*.SEZAINST data set into the system CLIST data set. Since the *hlq*.SEZAINST data set is in a fixed format, the SMTPNOTE member may be truncated if your system CLIST library is not in a fixed format.

You should customize the following variables in the SMTPNOTE CLIST:

HOSTNAME
: The name of the system on which this CLIST is installed. Typically, the name is the NJE node name of this system.

SMTPNODE
: The NJE node on which the SMTP server runs. Typically, HOSTNAME and SMTPNODE have the same value. When SMTPNODE is used on an NJE network in conjunction with a TCP-to-NJE gateway, the value of this parameter is the NJE node name of that gateway.

SMTPJOB
: The name of the address space in which SMTP runs at SMTPNODE. Usually this is SMTP.

TEMPDSN
: The name of the temporary data set used to store the contents of the note being created. This can be any arbitrary data set name that ends with the low-level qualifier, TEXT. Do not use a fully qualified name. If you do not fully qualify the name (no quotes), the data set name will be prefixed by the *userid*. If you enclose the name in single quotes, several users can use this temporary data set.

TIMEZONE
: The time zone for your system. This will appear in the ″Date:″ stamp of the RFC 822 header. See RFC 822 for valid time zone formats.

You should also modify the following PARMLIB members:

IEFSSNxx
: The IEFSSNxx member may be modified in one of the following two ways:

  - The following lines may be included:

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

where *nodename* is the NJE node name. The NJE node name, *nodename*, must be the same as the *hostname* and the *smtpnode* in the SMTPNOTE CLIST.

- If you are using restartable VMCF, you must make changes to IEFSSxx members in the SYS1.PARMLIB data set.

  For introductory information on restartable VMCF, refer to "Step 3: Configure VMCF and TNF" on page 56. For the MVS system changes required for restartable VMCF, refer to the TCP/IP for MVS Program Directory. For information on VMCF commands refer to *z/OS Communications Server: IP Diagnosis*.

IKJTSOxx
The TRANSREC statement must contain the correct nodename.

Step 4: Customize the SMTP Mail Headers (Optional)

Electronic mail has a standardized syntax for text messages that are sent across networks. The standard syntax is described in RFC 822. Messages have an envelope and contents. Envelopes contain all necessary information to accomplish transmission and delivery of the message content. The fields within the envelope are in a standard format.

In most cases, the mail header defaults are adequate. If it is necessary for you to change them, you can use the REWRITE822HEADER statement in the SMTP configuration data set to control the way SMTP performs a rewrite of the RFC 822 mail headers. Mail headers are passed from the local system, or NJE network, to the TCP network. Mail headers passing from the TCP network to the local system or NJE network are not affected. Only the addresses under certain RFC 822 header fields can be subject to the header rewriting rules.

The header fields affected by the REWRITE822HEADER statement are:

Field **Description**

From The identity of the person sending the message.

Resent-From
Indicates the person that forwarded the message.

Reply-To
Provides a mechanism for indicating any mailboxes to which responses are to be sent.

Resent-Reply-To
Indicates the person to whom you should forward the reply.

Return-Path
This field is added by the mail transport service at the time of final delivery. It contains definitive information about the address and route back to the originator of the message.

Sender
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 contents of this field are not included in copies sent to the primary and secondary recipients of the message but are included in the author's copy.

The SMTP Rules Data Set: You can override the default rules for header addresses by creating a SMTP rules data set. This allows you to customize the address transformations to the needs of a particular site. If you are customizing SMTP mail headers, this task is required.

The SMTP rules data set is pointed to by the //SMTPRULE DD statement in the SMTP cataloged procedure. The SMTP rules data set consists of:

Field Definition
Contains the names of all header fields whose addresses are to be rewritten

Rules Definition
Contains the rewrite rules for the header fields

Statement Syntax: In creating the SMTP rules data set you must use the following syntax conventions:

- The data set statements are free-format. Tokens can be separated by an arbitrary number of spaces, and statements can span an arbitrary number of lines. However, you must end every statement with a semicolon (;).
- A character string appearing within single quotation marks ('...') is not case-sensitive. For example, 'abc' represents 'abc', 'Abc', 'ABC', and so forth. The use of character strings is illustrated in the following sections.
- A character string appearing within double quotation marks ("...") is case-sensitive. For example, "abc" only represents "abc". It does not represent "Abc", "ABC", and so forth.

  Special characters, such as @ and % are treated the same whether enclosed by single quotation marks or double quotation marks.
- Double-hyphens ("--") are used to begin a comment. The comment extends to the end of the line.

The following sections describe the components of the SMTP rules data set.
- Format of the field definition section
- Format of the rules definition section

Format of the Field Definition Section: The field definition section is the first section in any SMTP rules data set. It defines any applicable alias fields, and it is introduced by the following heading:

```
Field Definition Section
```

This section allows similar fields to be grouped under an alias or common name. This name, or alias, is used to represent the field list. You can define an arbitrary number of aliases representing a set of field lists.

An alias name can be any alphanumeric sequence of characters that is not a predefined keyword within the SMTP rules (see the following). However, the alias name `DefaultFields` is treated specially by the SMTP configuration interpreter. If `DefaultFields` is defined, and if a rule is written that does not specify an associated field alias, the rules interpreter assumes that `DefaultFields` is the associated field alias.

The alias definition within this section is of the following form:

```
alias_name = alias_definition; optional comment
```

where *alias_name* is the name of the alias and *alias_definition* is an expression describing which fields are to be grouped under this alias. This expression can be as simple as a single field name. For example:

```
MyAlias = 'To';
```

The aliases can be a list or set of field names. The field names ***To, From, Cc***, and ***Bcc***, in the following example are part of a set of field names referenced by the alias `MyAlias`.

```
MyAlias = 'To' 'From' 'Cc' 'Bcc' ; -- first list of fields
```

You can combine field names and previously defined aliases to create a new alias. In the following example, the set of field names defined as `MyAlias` and the field names in the new alias `YourAlias` are combined to form a third set. The new alias `TheirAlias` is the union of both aliases and contains the fields of `MyAlias` and `YourAlias`.

```
MyAlias    = 'To' 'From' 'Cc' 'Bcc';
YourAlias  = 'Errors-To' 'Warnings-To';
TheirAlias =  MyAlias YourAlias;
```

In the previous example, `TheirAlias` is an alias that represents the following fields:

```
TheirAlias:  'To' 'From' 'Cc' 'Bcc' 'Errors-To' 'Warnings-To'
```

You can perform the following operations on set members of the alias to create a subset of the initial alias:

- Union operations
- Difference operations
- Intersection operations

Union and Difference Operations: Certain field names can be added to or omitted from a new alias of field names by using a minus sign to omit set members and an optional plus sign to include another field name. In the mathematics of sets, when you add together 2 or more sets, they form a union. When set members are omitted, the remaining set is created by the difference operation. In the following example `HerAlias` and `HisAlias` are defined. The alias `HisAlias` is created from the union of `TheirAlias`, `HerAlias`, and the omission of `Warning-To` and `Bcc` from the sets:

```
HerAlias   = 'Reply-To' 'Sender';
HisAlias   = TheirAlias - 'Warnings-To' - 'Bcc' + HerAlias;
```

In the previous example, `HisAlias` is an alias that represents the following fields:

```
HisAlias:  'To' 'From' 'Cc' 'Errors-To' 'Reply-To' 'Sender'
```

Intersection Operations: A field definition can include an intersection operation. When the intersection operation is applied to two field expressions, the resulting set contains the fields common to both. In the following example, `MyAlias` and `YourAlias` are defined. The alias `OurAlias` is created from the intersection of `MyAlias` and `YourAlias`. The asterisk (*) is the intersection operator.

```
MyAlias    = 'Bcc' 'Cc' 'From' 'Reply-To';
YourAlias  = 'Resent-From' 'Cc' 'Sender' 'To' 'Bcc';
OurAlias   = MyAlias * YourAlias; -- the intersection
```

In the previous example, `OurAlias` represents the following fields:

```
OurAlias:  'Bcc' 'Cc'
```

In the following complex example TheirAlias is created from the intersection of YourAlias with the sum of MyAlias plus Resent-From:

```
TheirAlias = (MyAlias + 'Resent-From') * YourAlias;
```

In the previous example, TheirAlias represents the following fields:

```
TheirAlias: 'Bcc' 'Cc' 'Resent-From'
```

The parentheses within the definition of TheirAlias perform the same functions as in algebra. Field expressions are evaluated from left to right, but the intersection operation has greater priority than union and difference operations. If parentheses were not used in the definition of TheirAlias, the result would be:

```
TheirAlias: 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
```

Format of the Rule Definition Section: The rule definition section is the next section in any SMTP RULES data set. It contains the header rewriting rules that define the intended address transformations, and it is introduced by the following heading:

```
Rule Definition Section
```

The basic form of a rewrite rule is:

alias :*before-address-pattern* => *after-address-pattern*;

where the alias name *alias* is an optional name representing the fields for which the rule is applicable. If the alias name *alias* : is omitted from this part of the rules, then DefaultFields is assumed.

The sequence of tokens that define how a particular type of address is to be recognized is the *before-address-pattern* portion of the rules definition. The sequence of tokens that define how the address is to appear after the address has been rewritten is the *after-address-pattern* portion of the rules definition. The following example is the rule for converting host names:

```
A '@' NJEHostName => A '@' TCPHostName; -- convert host names
```

In the previous example, A '@' NJEHostName is the *before-address-pattern* portion of this rule, and A '@' TCPHostName is the *after-address-pattern* portion. This rule specifies that the address to be rewritten has an arbitrary local name (A), an at-sign (@), and the NJE host name (NJEHostName) of the current site. The rule also specifies that the rewritten address must contain the same arbitrary local name (A), an at-sign, and the current site's TCP host name TCPHostName.

SMTP Rules Syntax Conventions: Use the following syntax convention when writing SMTP rules:

- Some keywords have special meaning to the rules interpreter. For example, NJEHostName keyword means the NJE host name of the present system, and TCPHostName keyword means the TCP host name of the present system. For more information about valid keywords see "Predefined Keywords within the SMTP Rules" on page 514. Some keywords, such as TCPHostName, have single values. Other keywords, such as AltTCPHostName and AnyDomainName, can have many possible values. To avoid ambiguity, any keyword that can have multiple values, and is used in the *after-address-pattern* of a given rule, must appear exactly once within the *before-address-pattern* of that rule. The following rule example shows a valid syntax:

  ```
  A '@' AltTCPHostName '.' AltTCPHostName =>
     A '%' TCPHostName '@' TCPHostName;
  ```

The following two rules have incorrect syntax because the first keyword `AltTCPHostName` must be rewritten to a keyword with specific values. The `AltTCPHostName` is attempting to be rewritten to the same `AltTCPHostName` but with unknown values, which is not valid.

```
A '@' AltTCPHostName '.' AltTCPHostName =>
   A '%' AltTCPHostName '@' TCPHostName;

A '@' TCPHostName => A '@' AltTCPHostName;
```

Any rule whose *before-address-pattern* includes a keyword that has a null value is ignored during the header rewriting. Thus, if there is no `AltNJEDomain` defined in the system configuration data set, no rule that includes `AltNJEDomain` in the *before-address-pattern* is considered during the header rewriting.

- Alphanumeric identifiers that are not within apostrophes or double quotation marks, and that are not predefined keywords, are considered wildcards in the rule statement. Wildcards represent an arbitrary (non-null) sequence of characters. The identifier `A`, in the previous rule example, is a wildcard. Thus, if `host` were the NJE host name for the current site, and if `tcphost` were the TCP host name for the current site, the previous rule example recognizes `abc@host` and `d@host` as candidates for address rewriting, and rewrites them as `abc@tcphost` and `d@tcphost` respectively. To avoid ambiguity, within the *before-address-pattern* of a given rule, no two wildcards are allowed in a row, and the same wildcard cannot be used more than once. The following rules have valid syntax:

  ```
  A '@' B TCPHostName      => A '%' B '@' TCPHostName;
  A '%' B '@' NJEHostName => A B '@' TCPHostName;
  ```

  The following rules have incorrect syntax because the first rule has 2 wildcards in a row `A` and `B`. The second rule has the same wildcard `A` repeated:

  ```
  A B '@' TCPHostName      => A A '%' B '@' TCPHostName;
  A '%' A '@' NJEHostName => A '@' TCPHostName;
  ```

- A character string appearing within apostrophes or double quotation marks tells the rules interpreter where a particular string is to appear within a header address. In the previous rule example, the `'@'` string in the *before-address-pattern* tells the rules interpreter that an at-sign (@) must appear between the arbitrary character string and the NJE host name. The `'@'` string in the *after-address-pattern* tells the rules interpreter that the address must be rewritten so an at-sign appears between the arbitrary string and the TCP host name. As previously mentioned, apostrophes denote strings that are not case-sensitive, and double quotation marks denote case-sensitive strings.
- The character sequence "=>", with no spaces between the characters, separates the *before-address-pattern* from the *after-address-pattern*.
- The order in which the rules are specified is important; the first rule encountered whose *before-address-pattern* matches the current address is the rule to dictate the address transformation. Once a matching rule has been found for an address, no other rule is considered.

In addition to the rules themselves, there is the capability for some simple logic to decide at system configuration time which rules within the data set should become active. These conditions are specified in the form of an IF-THEN-ELSE statement, as shown in the following example:

```
IF cond THEN
   statement list
ELSE
   statement list
ENDIF
```

A statement list can consist of any number of rules or nested IF statements, or both. Each IF statement, regardless of whether it is nested, must be ended by an ENDIF keyword. As with IF statements in other programming languages, the ELSE clause is optional.

There are only two conditions recognized by an IF statement:
1. IF *predefined keyword* = '*character string*' THEN ... ENDIF
2. IF *predefined keyword* CONTAINS '*character string*' THEN ... ENDIF

The conditional operators = and CONTAINS can be prefixed by the word NOT to invert the conditions.

The *predefined keyword* must be a keyword that resolves to a single value at system configuration time. The character string in the first condition can be null. A character string cannot span more than one line.

The following example shows the use of IF statements.

```
IF NJEDomain = ' THEN
   A '@' AnyNJEHostName => A '%' AnyNJEHostName '@' TCPHostName;
ELSE
   A '@' NJEHostName '.' NJEDomain    => A '@' TCPHostName;
   A '@' NJEHostName '.' AltNJEDomain => A '@' TCPHostName;
   IF NJEDomain CONTAINS '.' THEN
      A '@' AnyNJEHostName                   =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '@' AnyNJEHostName '.' NJEDomain;
   ELSE
      A '@' AnyNJEHostName                    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
   ENDIF
ENDIF
```

Predefined Keywords within the SMTP Rules: The following predefined keywords can be used to define the header rewriting rules:

AltNJEDomain
: Matches the alternative domain name of the NJE network as defined by the ALTNJEDOMAIN statement in the SMTP configuration data set.

AltTCPHostName
: Matches any alternative TCP host name of the system, as defined by ALTTCPHOSTNAME statements in the SMTP configuration data set.

AnyDomainName
: Matches any fully qualified domain name. Any host name with a period (.) is considered to be a fully qualified domain name.

AnyNJEHostName
: Matches any (unqualified) NJE host name defined in the SMTPNJE.HOSTINFO data set.

NJEDomain
: Matches the domain name of the NJE network as defined by the NJEDOMAIN statement in the SMTP configuration data set.

NJEHostName
Matches the NJE host name of the system.

SecureNickAddr
Matches an address of the form *NJE_user_id@NJE_node_id*, where *NJE_user_id*, and *NJE_node_id* are defined with a nickname in the SMTP security data set.

Note: This only matches user and node IDs that are defined with nicknames.

When SecureNickAddr is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickName with the corresponding nickname. This allows SecureNickName to be specified in the *after-address-pattern*.

SecureNickName
Matches a nickname defined in the SMTP security data set. When SecureNickName is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickAddr with the corresponding *NJE_user_id@NJE_node_id*. This allows SecureNickAddr to be specified in the *after-address-pattern*.

ShortTCPHostName
Matches the first portion of the TCP host name of the system, as defined by the HOSTNAME statement in the *hlq*.TCPIP.DATA data set. For example, if the TCP host name was `mvs1.acme.com`, the value of ShortTCPHostName is `mvs1`.

TCPHostName
Matches the TCP host name of the system as defined by the concatenation of the HOSTNAME and DOMAINORIGIN statements in the *hlq*.TCPIP.DATA data set.

TCPHostNameDomain
Matches the domain portion of the TCP host name of the system as defined by the DOMAINORIGIN statement in the *hlq*.TCPIP.DATA data set. For example, if the TCP host name was `mvs1.acme.com`, the value of TCPHostNameDomain is `acme.com`.

The predefined keywords can consist of any combination of uppercase and lowercase characters; the rules interpreter does not distinguish between them.

The secure keywords are only valid when SMTP is configured to be a secure gateway.

Default SMTP Rules: If the //SMTPRULE DD statement is not found, SMTP uses a default set of rules. The default set used depends on whether SMTP is configured as a secure gateway.

SMTP Nonsecure Gateway Configuration Defaults: If SMTP is not configured as a secure gateway, SMTP uses the following defaults:

```
Field Definition Section

DefaultFields = 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
                'Resent-Reply-To' 'Resent-Sender' 'Return-Path'
                'Sender' 'To';

Rule Definition Section
```

```
A '@' NJEHostName => A '@' TCPHostName;

IF NJEDomain = ' THEN
   A '@' AnyNJEHostName => A '%' AnyNJEHostName '@' TCPHostName;
ELSE
   A '@' NJEHostName '.' NJEDomain    => A '@' TCPHostName;
   A '@' NJEHostName '.' AltNJEDomain => A '@' TCPHostName;
   IF NJEDomain CONTAINS '.' THEN
      A '@' AnyNJEHostName                  =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '@' AnyNJEHostName '.' NJEDomain;
   ELSE
      A '@' AnyNJEHostName                   =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
   ENDIF
ENDIF

A '@' TCPHostName      => A '@' TCPHostName;
A '@' ShortTCPHostName => A '@' TCPHostName;
A '@' AltTCPHostName   => A '@' TCPHostName;
A '@' AnyDomainName    => A '@' AnyDomainName;
A '@' B                => A '@' B '.' TCPHostNameDomain;
```

SMTP Secure Gateway Configuration Defaults: If SMTP is configured as a secure gateway, SMTP uses the following defaults:

```
Field Definition Section

DefaultFields = 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
                'Resent-Reply-To' 'Resent-Sender' 'Return-Path'
                'Sender' 'To';

Rule Definition Section

SecureNickAddr     => SecureNickName '@' TCPHostName;
A '@' NJEHostName => A '@' TCPHostName;

IF NJEDomain NOT = ' THEN
   SecureNickAddr '.' NJEDomain    => SecureNickName '@' TCPHostName;
   SecureNickAddr '.' AltNJEDomain => SecureNickName '@' TCPHostName;
   A '@' NJEHostName '.' NJEDomain    => A '@' TCPHostName;
   A '@' NJEHostName '.' AltNJEDomain => A '@' TCPHostName;
   IF NJEDomain CONTAINS '.' THEN
      A '@' AnyNJEHostName                  =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '@' AnyNJEHostName '.' NJEDomain;
   ELSE
      A '@' AnyNJEHostName                   =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
   ENDIF
ENDIF
```

```
A '@' TCPHostName      => A '@' TCPHostName;
A '@' ShortTCPHostName => A '@' TCPHostName;
A '@' AltTCPHostName   => A '@' TCPHostName;
A '@' AnyDomainName    => A '@' AnyDomainName;
A '@' B                => A '@' B '.' TCPHostNameDomain;
```

Examples of Header Rewrite Rules: The following examples show how the header rewriting rules affect an SMTP mail header. The example site is not a secure gateway and is configured as follows:

```
TCPHostName      = mvs1.acme.com
ShortTCPHostName = mvs1
AltTCPHostName   = seeds.acme.com
NJEHostName     = mvs1
NJEDomain       = acmenet
AltNJEDomain    = centralnet
```

Note that the above keywords are configured according to the definitions found in "Predefined Keywords within the SMTP Rules" on page 514 (for example, from *hlq*.TCPIP.DATA). In addition, assume that the following are known to be other NJE hosts:

```
bird
iron
```

Then the following header:

```
From: abc@mvs1 (Brendan Beeper)
To: Jenny Bird <def@bird>
Cc: ghi@iron.acmenet, j@mvs1,
 k@seeds.acme.com,
 Mailing List <owner@acmenet>,
 lmno@iron.centralnet
Subject: New Ore
```

is rewritten by the default header rewriting rules as:

```
From: abc@mvs1.acme.com (Brendan Beeper)
To: Jenny Bird <def%bird.acmenet@mvs1.acme.com>
Cc: ghi%iron.acmenet@mvs1.acme.com, j@mvs1.acme.com,
 k@mvs1.acme.com,
 Mailing List <owner%acmenet@mvs.acme.com>,
 lmno%iron.acmenet@mvs1.acme.com
Subject: New Ore
```

The next example deviates from the defaults listed in "Default SMTP Rules" on page 515. On the configuration for nonsecure gateways, if you change the rule before the 2 ENDIFs to:

```
A '@' AnyNJEHostName '.' AltNJEDomain =>
   '<@' TCPHostName ':' A '@' AnyNJEHostName '.' NJEDomain '>';
```

then the last address in the Cc: field within our header is rewritten as:

```
Cc: <@mvs1.acme.com:lmno@iron.acmenet>
```

Note: Do not make the change shown in the previous example; it is intended only as a demonstration of the capabilities of the pattern-matching language.

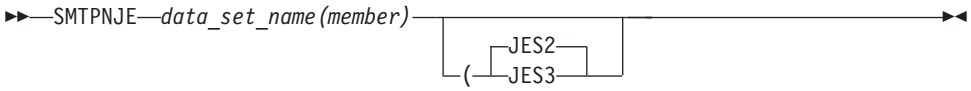
Step 5: Set Up a TCP-to-NJE Mail Gateway (Optional)

You can configure the SMTP server to run as a mail gateway between TCP network users and users located on a NJE network attached to the local host. This way NJE users can send mail or data sets to users on TCP hosts using SMTPNOTE. See *z/OS Communications Server: IP User's Guide* for more information about

SMTPNOTE. For JES2, see *MVS/ESA JES2 Initialization and Tuning* , SC28-1038-1. For JES3, see *MVS/ESA JES3 Initialization and Tuning* , SC23-0073-2.

Follow these steps to set up your TCP-to-NJE mail gateway:

1. Add the GATEWAY statement to the SMTP configuration data set. Add other related statements, such as ALTDOMAIN, NJECLASS, NJEDOMAIN, and NJEFORMAT, as required by your configuration.
2. Issue the SMTPNJE command.



data_set_name(member)
The name of the input data set for SMTPNJE. It specifies the initialization data set of the JES2 or JES3 subsystem that is scanned for NJE nodes by SMTPNJE. The data set name is the same name as defined on ddname HASPPARM in your JES2 procedure or in the JES3IN ddname in your JES3 procedure.

member is the JES2PARM member that contains the NODE and DESTID entries for your installation.

(Required delimiter.

JES2 or JES3
Denotes whether the initialization data set being pointed to is for JES2 or JES3. If omitted, the default is JES2. For JES2, the SMTPNJE program scans for the keywords NODE and DESTID from which it extracts the information. For JES3, the keyword scanned for is NJERMT.

The SMTPNJE program creates the NJE host table data set called *user_id*.SMTPNJE.HOSTINFO. You can rename this data set and include the name of the data set on the SMTPNJE DD statement in the SMTP cataloged procedure. The //SMTPNJE DD statement is required.

3. Install the SMTP server (along with the TCPIP address space) on the gateway node. Use the GATEWAY, NJEDOMAIN, and NJEFORMAT statements in the configuration data set. Optionally, you can use either the RESTRICT or the SECURE statements to limit which users can use the gateway.

Step 6: Specify Configuration Statements in SMTP Configuration Data Set

Copy the member *hlq*.SEZAINST(SMTPCONF) to your own SMTP configuration data set and modify it for your site using the SMTP configuration statements.

Note: If the SMTP configuration data set is a sequential data set, you cannot edit the data set while SMTP is running. If the data set is a PDS member, it can be edited while SMTP is running.

Summary of SMTP Configuration Statements: The SMTP configuration statements are summarized in Table 20.

Note: Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these statements.

Table 20. Summary of SMTP Configuration Statements

Statement	***Description***
ALTNJEDOMAIN	Specifies an alternative domain name of the NJE network, if SMTP is running as a mail gateway.
ALTTCPHOSTNAME	Specifies an additional host name for the local host. Mail received for this host name is accepted and delivered locally.
BADSPOOLFILEID	Specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail.
CHECKSPOOLSIZE	Enables SMTP to check the size of the JES spool file prior to writing the data to the hlq.TEMP.NOTE file.
DBCS	Specifies that DBCS code conversion be performed on the mail.
DEBUG	Records all SMTP commands and replies.
FINISHOPEN	Specifies the SMTP wait time for connection.
GATEWAY	Specifies operation of SMTP as a gateway.
INACTIVE	Specifies the SMTP wait time before closing an inactive connection.
IPMAILERADDRESS	Specifies the IP address of an SMTP server that can resolve network addresses of unknown hosts.
LISTENONADDRESS	Allows you to restrict which IP address is used to receive mail on a multi-homed system.
LOCALCLASS	Specifies the spool data set class for local mail delivery.
LOCALFORMAT	Specifies the spool data set format for local host mail delivery.
LOG	Directs SMTP to log all SMTP traffic.
MAILER	Specifies the address of the batch SMTP server that receives mail.
MAILFILEDSPREFIX	Specifies the prefix to add to mail data sets.
MAILFILESUNIT	Specifies the unit where SMTP mail data sets reside.
MAILFILEVOLUME	Specifies the volume where newly allocated SMTP data sets reside.
MAXMAILBYTES	Specifies the maximum size of mail that is accepted over a TCP connection.
NJECLASS	Specifies the spool data set class for mail delivered on an NJE network.
NJEDOMAIN	Specifies the domain name of the NJE network if SMTP functions as a gateway.
NJEFORMAT	Specifies the spool data set format for mail delivered on the NJE network.
NJENODENAME	Specifies the node name of the local JES2 or JES3 node for mail delivered on the NJE network.
NOLOG	Turns off the logging of mail transactions.
OUTBOUNDOPENLIMIT	Specifies a limit on the maximum number of simultaneous TCP connections over which SMTP actively delivers mail.
PORT	Specifies an alternative port number for the SMTP server during testing.
POSTMASTER	Specifies the address (or addresses) for mail addressed to the postmaster at the local host.
RCPTRESPONSEDELAY	Specifies how long the SMTP server delays responding to the RCPT commands.
RESOLVERRETRYINT	Specifies the number of minutes SMTP waits between attempts to resolve domain names.
RESTRICT	Specifies addresses of users who are not allowed to use SMTP mail services.
RETRYAGE	Specifies the number of days after which mail is returned as undeliverable.

Table 20. Summary of SMTP Configuration Statements (continued)

Statement	Description
RETRYINT	Specifies the number of minutes between attempts to send mail to an inactive TCP host.
REWRITE822HEADER	Prevents SMTP from rewriting RFC822 headers with source routing.
SECURE	Specifies that SMTP operates as a secure mail gateway between TCP network sites and NJE network sites.
SMSGAUTHLIST	Specifies the addresses of users authorized to issue privileged SMTP SMSG commands.
SPOOLPOLLINTERVAL	Specifies the interval for SMTP to check the spool for incoming batch data sets.
TEMPERRORRETRIES	Specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem.
TIMEZONE	Sets the printable name of the local time zone.
WARNINGAGE	Specifies the number of days after which a copy of the mail is returned to the sender, indicating that the mail has so far been undeliverable and that SMTP will continue to retry delivery for RETRYAGE days.

Sample SMTP Configuration Data Set (SMTPCONF): The following sample is found in *hlq*.SEZAINST(SMTPCONF).

```
;*********************************************************************
;                                                                    *
;   Name of Data Set:    SMTP.CONFIG                                 *
;                                                                    *
;   COPYRIGHT = NONE.                                                *
;                                                                    *
;   This data set is pointed to by the CONFIG DD statement in the    *
;   SMTP Cataloged Procedure (SMTPPROC).                             *
;                                                                    *
;   This data set is used to specify runtime options and data        *
;   to the SMTP server address space.                                *
;                                                                    *
;   Syntax Rules for the SMTP configuration data set:                *
;                                                                    *
;   (a) All characters to the right of and including a ; will be     *
;       treated as a comment.                                        *
;                                                                    *
;   (b) Blanks and <end-of-line> are used to delimit tokens.         *
;                                                                    *
;   See the TCP/IP for MVS:  Customization and Administration Guide  *
;   for a complete explanation of all the statements.                *
;                                                                    *
;*********************************************************************
;
;  Defaults that normally aren't changed:
;
PORT 25                 ; Port to accept incoming mail
BADSPOOLFILEID TCPMAINT ; Mailbox where unreadable spool files and
                        ; looping mail are transferred.
LOG                     ; Log all SMTP mail delivered
INACTIVE 180            ; Time-out for inactive connections
FINISHOPEN 120          ; Time-out for opening TCP connections
RETRYAGE 3              ; Keep retrying mail delivery for 3 days
WARNINGAGE 1            ; Warn sender that mail has been undeliverable
                        ; for 1 day, but that attempts to deliver the
                        ; mail will continue for another 2 days.
RETRYINT 20             ; Retry mail delivery every 20 minutes
MAXMAILBYTES 524288     ; Largest mail to accept over a TCP connection
RESOLVERRETRYINT 20     ; Retry pending name resolutions every 20 minutes
```

```
RCPTRESPONSEDELAY 60    ; How long to delay RCPT TO: response when
                        ; waiting for an address resolution.
TEMPERRORRETRIES 0      ; How many times to retry temporary delivery
                        ; errors.  The default, 0, means retry for
                        ; RETRYAGE days; otherwise the mail is returned
                        ; after this number of deliver attempts.
SPOOLPOLLINTERVAL 30    ; Amount of time in seconds between spool polling
TIMEZONE EST            ; Specifies the printable 3-letter name of
                        ; the local time zone.  Remember to change this
                        ; for daylight saving time.
; DEBUG                 ; Normally not used, causes debug information to
                        ; be written to the SMTP DEBUG file.
;
;*********************************************************************
;
; The following statements tell SMTP where incoming mail is stored
; while it is being queued for delivery.
;
MAILFILEDSPREFIX SMTP   ; Data set prefix name for queued mail files
MAILFILEUNIT SYSDA      ; MVS unit name for new file allocations
; MAILFILEVOLUME volume ; MVS volume name for new file allocations
;
;*********************************************************************
;
; ALTTCPHOSTNAME is used to specify an alternative fully qualified host
; name by which SMTP will know the local host.  Mail sent to users at
; <hostname> is treated as if they were local users.
;
; ALTTCPHOSTNAME <hostname>
;
;*********************************************************************
;
; The POSTMASTER statement specifies the mailboxes where mail
; addressed to postmaster is spooled.  Multiple POSTMASTER statements
; can be specified when not running in SECURE mode.
;
POSTMASTER TCPMAINT
; POSTMASTER NJEuser@NJEnode
; POSTMASTER SMTPuser@SMTPnode
;
;*********************************************************************
;
; Use the SMSGAUTHLIST statement to specify the addresses of local
; users authorized to issue privileged SMTP SMSG commands.
;
SMSGAUTHLIST
    TCPMAINT
  ; OPERATOR  LocalUser
ENDSMSGAUTHLIST
;
;*********************************************************************
;
; The OUTBOUNDOPENLIMIT statement specifies the maximum number of
; simultaneous TCP connections over which SMTP will actively deliver
; mail.
;
; OUTBOUNDOPENLIMIT 30
;
;*********************************************************************
;
; Configuration for a typical NJE to TCP/IP mail gateway.
;
GATEWAY               ; Accept mail from and deliver mail to NJE hosts
NJEDOMAIN BITNET      ; Pseudo domain name of associated NJE network
NJEFORMAT PUNCH       ; NJE recipients receive mail in punch format
NJECLASS B            ; spool class for mail delivered by SMTP to the
                      ; NJE network
```

```
LOCALFORMAT NETDATA    ; Local recipients get mail in netdata format
                       ; Netdata allows TSO receive to be used with mail
LOCALCLASS B           ; Spool class for local mail delivered by SMTP
;
;*********************************************************************
;
; Use the ALTNJEDOMAIN to specify an alternate NJE domain name.
; This can be useful when the NJE network is known by multiple
; domain names, such as "vnet" and "vnet.ibm.com".
;
; ALTNJEDOMAIN vnet
;
;*********************************************************************
;
; The REWRITE822HEADER statement specifies whether SMTP will
; rewrite the RFC822 headers on all mail passing from NJE to TCP
; through the mail gateway.  The SMTP.RULES data set specifies how the
; server is to rewrite the headers.
;
; The PRINT | NOPRINT options specify whether SMTP will print
; the rules to the SMTP output when SMTP starts.
;
REWRITE822HEADER YES NOPRINT
;
;*********************************************************************
;
; Use MAILER to specify the address of a batch SMTP server to which
; SMTP delivers mail destined for various classes of recipients.  The
; old FOLDnoSOURCEroute parameter is equivalent to specifying PUNCH and
; NOSOURCEROUTES.
;
; LOCAL, NJE, and UNKNOWN specify conditions under which mail will
; be forwarded to the MAILER - see the Customization and Administration
; Guide for further details.
;
; MAILER MUSER@MNODE PUNCH NOSOURCEROUTES LOCAL NJE UNKNOWN
;
; MAILER ... UNKNOWN and IPMAILERADDRESS should not be used together.
;
; IPMAILERADDRESS x.xx.xxx.xx ; Routes mail sent to an unknown
;                      ; recipient to an SMTP server on an IP network
;
;*********************************************************************
;
; The RESTRICT statement specifies addresses of users who cannot
; utilize SMTP services.
;
; RESTRICT RETURN          ; Return mail from restricted users
;   charming@ourvm.our.edu ; Don't accept any mail from Prince Charming
;   charming@OURVMX        ; via NJE or TCP network.
;   charming@ourvm*        ; This line takes the place of previous 2 lines!
;  *@castle                ; Don't accept mail from anyone at host castle
; ENDRESTRICT
;
;*********************************************************************
;
; Use the SECURE statement if this SMTP machine is to run as an SMTP-
; to-NJE Secure Gateway.  Only users in the SMTP.SECTABLE data set
; will be allowed to send mail, all other mail will be returned or
; rejected.  Note that the contents of dataset
; mailfiledsprefix.SECURITY.MEMO will be sent to NJE users that are
; not authorized to use the gateway.
;
; SECURE
;
;*********************************************************************
;
```

```
; Use the DBCS statement if this SMTP machine is to perform DBCS
; code conversion on the mail.  Consult the Customization and
; Administration Guide for the necessary parameters.
;
; DBCS   <parameters>
;
;
```

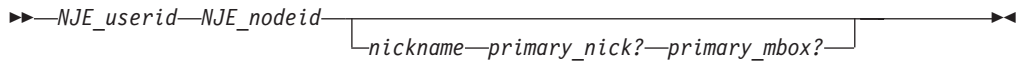
Step 7: Create an SMTP Security Table (Optional)

If you want to set up a secure TCP-to-NJE gateway, you need to:

- Include the SECURE statement in the SMTP configuration data set
- Create a security data set that contains a list of NJE users who are authorized to use the gateway
- Create a mailfiledsprefix.SECURITY.MEMO data set. The contents data set are sent to unauthorized NJE users whose mail is rejected. In this section, see Rejected Mail Examples for sample contents of this data set. This data set must be defined as LRECL=255 and RECFM=VB. It will be dynamically allocated by SMTP when needed.

The SMTP security data set is pointed to by //SECTABLE DD statement. The security table data set must be allocated with LRECL=255 and RECFM=VB. Records whose first nonblank character is an asterisk (*) are treated as comments and are ignored.

Use the following format when creating the list of NJE users:



NJE_userid
The NJE user ID of the authorized user.

NJE_nodeid
The NJE node ID of the authorized user.

nickname
The name by which this user is known on the TCP side of the gateway. This name must not contain any special characters, such as < > ( ) [ ] \ . , ; : @ and ".

primary_nick?
Either Y or N. If Y is specified, then mail addressed to *nickname*`@smtp-gateway` is automatically forwarded to *NJE_userid* `at` *NJE_nodeid*. Each nickname can have only one *primary_nick?* record set to Y.

primary_mbox?
Either Y or N. If Y is specified, then mail from *NJE_userid* `at` *NJE_nodeid* is converted to *nickname*`@smtp-gateway` before it is sent to the TCP recipient. Each *NJE_userid*, *NJE_nodeid* pair can only have one *primary_mbox?* record.

SMTP Security Data Set Examples: The following example shows an SMTP security data set:

```
*  Records for Jane Doe, within IBM
JDOE   ALMADEN
JDOE   AUSTIN
*  Records for John Smith, within IBM
```

```
SMITH  RALEIGH   JOHNNY  Y  N
SMITH  YORKTOWN  JOHNNY  N  Y
SMITH  DALLAS    JOHNNY  N  N
SMITH  RALEIGH   JSMITH  Y  Y
```

For example, mail sent from the following NJE network addresses through the SMTP gateway is rewritten to the following TCP network addresses. Assume the host name of the gateway is SMTP-GATEWAY.IBM.COM.

```
NJE Address              TCP Address
-------------------------------------------------------------------------

JDOE  at ALMADEN         JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM
JDOE  at AUSTIN          JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM
SMITH at RALEIGH         JSMITH@SMTP-GATEWAY.IBM.COM
SMITH at YORKTOWN        JOHNNY@SMTP-GATEWAY.IBM.COM
SMITH at DALLAS          JOHNNY%DALLAS@SMTP-GATEWAY.IBM.COM
```

Mail sent from the TCP network to the following TCP network addresses is forwarded to the following NJE network addresses. Assume the host name of the gateway is SMTP-GATEWAY.IBM.COM.

```
TCP Address                                NJE Address
-------------------------------------------------------------------------

JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM          JDOE  at ALMADEN
JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM           JDOE  at AUSTIN
JSMITH@SMTP-GATEWAY.IBM.COM                SMITH at RALEIGH
JOHNNY@SMTP-GATEWAY.IBM.COM                SMITH at RALEIGH
SMITH%DALLAS@SMTP-GATEWAY.IBM.COM          SMITH at DALLAS
```

Rejected Mail Examples: SMTP rejects mail to or from an unauthorized NJE user. If the mail is from the TCP network, SMTP rejects the RCPT TO command with the error:

```
550 User is not a registered gateway user
```

If the mail is from the NJE network, SMTP rejects the MAIL FROM command with the error:

```
550 User is not a registered gateway user
```

and includes the *mailfiledsprefix*.SECURITY.MEMO data set as an explanation.

The following example shows a sample *mailfiledsprefix*.SECURITY.MEMO data set:

```
The mail you sent to this SMTP gateway cannot be delivered because
you are not a registered user of this gateway.  Contact your local
administrator for instructions on how to be authorized to use this
SMTP gateway.
```

The following is an example of rejected mail that was sent to an unregistered NJE user:

```
Date: Fri, 5 Jul 91 10:55:59 EST
From: SMTP@MVS1.ACME.COM
To: DANIEL@MVS1
Subject: Undeliverable Mail

MVS1.ACME.COM unable to deliver following mail to recipient(s):
    <MATT@SMTP-GATEWAY.IBM.COM>
MVS1.ACME.COM received negative reply from host:
    SMTP-GATEWAY
550 User 'MATT@SMTP-GATEWAY' is not a registered gateway user

           ** Text of Mail follows **
```

```
Date: Fri, 5 Jul 91 10:55:56 EDT
From: <DANIEL@MVS1.ACME.COM>
To:   <MATT@SMTP-GATEWAY.IBM.COM>
Subject:  Lunch
Matt,

Do you have time to meet for lunch next week?  I want to discuss the
shipment of ACME iron birdseed.
Daniel
```

The following is an example of rejected mail that was sent from an unregistered NJE user:

```
Date: Fri, 5 Jul 91 11:35:18 EST
From: <SMTP@SMTP-GATEWAY.IBM.COM>
To: <MATT@SMTP-GATEWAY.IBM.COM>
Subject: Undeliverable Mail
Unable to deliver mail to some/all recipients.
050 MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
550-User 'MATT@SMTP-GATEWAY' is not a registered gateway user.
550-
550-The mail you sent to this SMTP gateway cannot be delivered because
550-you are not a registered user of this gateway.  Contact your local
550-administrator for instructions on how to be authorized to use this
550 SMTP gateway.
           ** Text of Mail follows **
HELO SMTP-GATEWAY.IBM.COM
MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
RCPT TO:<DANIEL@MVS1.ACME.COM>
DATA
Date: Fri, 5 Jul 91 11:34:17 EST
From: <MATT@SMTP-GATEWAY.IBM.COM>
To:   <DANIEL@MVS1.ACME.COM>
Subject:  Awaiting your message
Daniel,
When are you going to contact me about the iron birdseed and giant
electromagnet that I ordered?  I would like to meet with you soon.
Matt

.
QUIT
```

Step 8: Enable SMTP Domain Name Resolution

The SMTP server can be configured to use either a domain name server or the local site tables. The *hlq*.TCPIP.DATA data set must be configured before you can use the domain name resolution for SMTP.

To use a domain name server, configure the *hlq*.TCPIP.DATA data set with the IP address of one or more name servers. If the *hlq*.TCPIP.DATA data set does not point to any name servers, the local site tables are used. However, if the SMTP server is configured to use name servers, SMTP does not use the site tables.

When SMTP uses a domain name server, it asks the domain name server for the MX records for the host to which it is trying to connect. If SMTP does not find MX records for a host, it delivers mail only to the primary host listed in the A records. The MX and A records are coded in the domain name server database.

The basic idea behind MX records is to send the mail as close as possible to the final destination. The destination host may currently be inactive, for example, because it is in another time zone. SMTP needs a synchronous connection to deliver the mail, but due to the different time zones, two systems might never be active at the same time and would never be able to exchange mail.

Using MX records would allow the SMTP server to deliver the mail to an alternate host if the first one is unavailable. SMTP tries to deliver mail to the host with the lowest MX record count. If the host is not currently available, it tries the host with the next lowest count.

For example, if SMTP wants to send mail to USER@BASKET, it checks the name server for MX records and finds the following:

```
MVS20  BASKET  A
       BASKET  MX  0    MVS20
       BASKET  MX  5    MVS18
       BASKET  MX  10   VMQ
```

SMTP delivers the mail to the BASKET with the lowest count on its MX record. If `MVS20` is unable to receive the mail, SMTP then tries to deliver it to `MVS18`. If `MVS18` cannot receive the mail, it tries `VMQ`. If none of the hosts can receive the mail, SMTP stores the mail and queues it for later delivery, at which time the process repeats.

For more information about how to add MX records to your name server, consult RFC 974, “Mail Routing and the Domain System.”

To receive a detailed trace on how SMTP is resolving a particular host name, you can issue the SMSG SMTP TRACE command at the console. You can also add the TRACE RESOLVER statement when configuring the *hlq*.TCPIP.DATA data set, but this will also trace the name resolution for all the other applications using the name server. To prevent the console log from becoming too large, only use the TRACE RESOLVER statement for debugging.

If changes to the domain name server requires you to re-resolve already-queued mail, use the SMSG SMTP EXPIRE command as described in the *z/OS Communications Server: IP User’s Guide*. You can also query operating statistics, such as mail delivery queues of the SMTP server, by using the SMSG SMTP command. This and other administrative tasks are discussed in more detail in the *z/OS Communications Server: IP User’s Guide*.

Step 9: Enable Sending Messages to SMTP Users and Users on an IP Network

The SMTP server can be configured to send ALL your TCP/IP SMTP mail to a specified mail server, or mail relay. You may need to do this if you have installed a FIREWALL.

This is accomplished by using the IPMAILERADDRESS statement in the *hlq*.SMTP.CONFIG file.

To utilize this technique for all mail, perform the following steps:

1. Inhibit SMTP from resolving hostname via domain name server specified in the *hlq*.TCPIP.DATA file.
   a. Create a new *hlq*.TCPIP.DATA file based on your original one.
   b. Remove any NSINTERADDR statements.
   c. Modify the //SYSTCPD DD statement to your SMTP startup up procedure to point to this modified *hlq*.TCPIP.DATA file.
2. Inhibit SMTP from resolving hostname via search of the local hosts file.
   a. Create an empty HOST.LOCAL file (for use by SMTP).
   b. From this file, create corresponding files: SMTP.HOSTA.ADDRINFO and SMTP.HOSTS.SITEINFO.

Note: It is assumed the job name of SMTP is SMTP. If you use a different job name, then SMTP needs to be the name of the SMTP job.

3. Update the SMTP.CONFIG file to redirect mail to a specific server using the IPMAILERADDRESS statement:

```
IPMAILERADDRESS ip_address
```

   where ip_address is address of the mail server that can perform the hostname resolution.

   For more information, refer to *z/OS Communications Server: IP Configuration Reference*.

Configuring z/OS UNIX sendmail and Popper

The following is intended to provide the administrator with specific information on how to configure sendmail on the z/OS platform. Before using this chapter, become familiar with the industry-accepted publication for sendmail, *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-222-0). That publication is known throughout the industry as simply the *"bat book"*, and this chapter consistently refers to the *"bat book"* for further information.

Additional information about sendmail can also be found on the z/OS UNIX application website, http://www.s390.ibm.com/unix, as well as in documents from the sample directory that were received during the port of sendmail 8.8.7 from the http://www.sendmail.org website. The Sendmail Installation and Operation Guide document (sendmail.ps), for instance, is the generic guide from http://www.sendmail.org, which might be helpful as a more thorough guide in a slightly different format. The README.m4 document gives more details for building a configuration file using the m4 preprocessor.

This chapter also provides information on how to configure popper on the z/OS platform. The popper function requires very little configuration. For more information on the protocol used by this UNIX application, see RFC1939.

Overview

The simple mail architecture in which sendmail and popper fit includes a mail user agent (MUA), a mail transfer agent (MTA), and a mail delivery agent (MDA). An MUA is client software that a user invokes directly to send and receive e-mail. Examples of MUAs include Eudora, Netscape Navigator, pine and elm. An MTA is software that actually routes messages from a sender's system to the receiver's system. sendmail is an MTA. It's worth noting, however, that sendmail relies on other programs to implement non-SMTP based transport (for example, UUCP-based transport as well as local delivery to a user's mail spool file). An MDA is server software that delivers received mail to a user's MUA. Popper is an example of an MDA using the POP3 protocol.

At the sender's end of the mail delivery process, the sender's MUA transmits the message to be delivered to sendmail, as shown in Figure 57 on page 528.

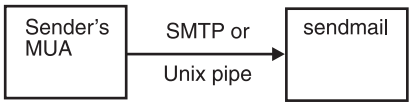


Figure 57. Sender MUA Transmits the Message to sendmail

This can occur in one of two ways. If the MUA is running on the local host, the message can be transmitted by executing a copy of sendmail and transmitting the message to the standard input of that process via a UNIX pipe.

Alternatively (and more commonly), a copy of sendmail will be running as a daemon, and the MUA (running on either the local host, or on a remote host) will open an SMTP connection to the sendmail daemon, transmitting the message to be delivered via that SMTP connection. In this case, sendmail is acting as an SMTP server, while the MUA is acting as an SMTP client.

In the next step, for each recipient address, sendmail transmits the message to some other SMTP server, to route the message to its final destination at the recipient's site. This is shown in Figure 58.

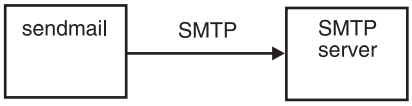


Figure 58. sendmail Transmits the Message to an Intermediate SMTP Server

The receiving SMTP server, in this case, might be a local hub that handles all mail at the sender's site, a remote hub handling all mail at the recipient's site, or an SMTP server at the recipient's host system.

In the next step, sendmail acts as an SMTP client, initiating an SMTP connection with some SMTP server, and then transmitting the message to be delivered to that server, via the SMTP connection.

At the receiver's end of the mail delivery process, a sendmail daemon receives the message from some SMTP client, as shown in Figure 59.

Figure 59. A sendmail Daemon Receives the Message from an SMTP Client

The sendmail daemon, acting as an SMTP server, accepts an incoming SMTP connection, and receives a message to be delivered over that SMTP connection. (This is identical to receipt of a message from an MUA, over an SMTP connection.)

Upon receiving the message, sendmail delivers it to the local recipient by appending the message to the recipient's mail spool file. To do this, sendmail requires a local mailer program, as depicted in Figure 60 on page 529.

Figure 60. Sendmail Delivers the Message to the Local Recipient

In this step, sendmail executes a specified local mailer program, such as /bin/mail, and transmits the message to be delivered to that mailer via a Unix pipe. The mailer program appends the message to the recipient's mail spool file. With this sendmail's role in delivery of mail is completed.

For the recipient to now read the received message, an MUA must be used. As mentioned in the previous section, depending upon the MUA, this may or may not require an additional MDA, such as popper. If the receiver's MUA has direct access to the mail spool file, the MUA may retrieve the mail directly from the spool file, as depicted in Figure 61.

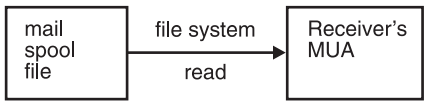


Figure 61. Receiver's MUA has Direct Access to the Mail Spool File

Alternatively (and more commonly), the MUA will establish a POP3 connection with a popper daemon, and retrieve the message over that connection. This is shown in Figure 62.

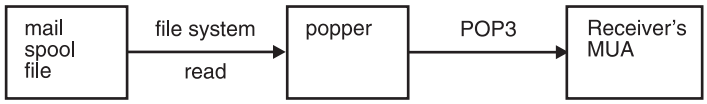


Figure 62. Receiver's MUA Retrieves the Message over a POP3 Connection with a Popper Daemon

The popper daemon will also allow the receiver's MUA to manage the mail spool file, by allowing it to specify whether and which message should be deleted.

Configuring z/OS UNIX sendmail

This section contains information on the following:

- The sendmail samples directory
- Creating the configuration file
- Creating an aliases file
- Configuration hints and tips

The sendmail Samples Directory

Much of the sendmail samples directory is dedicated to the automated creation of the configuration file. The /usr/lpp/tcpip/samples/sendmail/cf directory contains a sample.mc file and the subsequent sample.cf configuration file that was created by running the m4 macro preprocessor on the sample.mc file. If the /usr/lpp/tcpip/samples/sendmail directory is examined, the following directory structure can be found:

```
cd /usr/lpp/tcpip/sample/sendmail
ls
README.m4    feature   mailer      siteconfig
cf           hack      ostype      sendmail.ps
domain       m4        sh
```

cf
: Both site-dependent and site-independent descriptions of hosts. Files ending in **.mc** (″Master Configuration″) are the input descriptions. The output is in the corresponding **.cf** file. The general structure of these files is described below.

domain
: Site-dependent subdomain descriptions. These are tied to the way your organization wants to do addressing. These descriptions are referenced using the DOMAIN *m4* macro in the **.mc** file.

feature
: Definitions of specific features that some particular host in your site might want. These are referenced using the FEATURE *m4* macro. An example feature is `use_cw_file`, which tells z/OS UNIX sendmail to read an `/etc/sendmail.cw` file on startup to find the set of local names.

hack
: Local hacks, referenced using the HACK *m4* macro. Avoid these.

m4
: Site-independent *m4(1)* include files that have information common to all configuration files. Think of this as a ″#include″ directory.

mailer
: Definitions of mailers, referenced using the MAILER *m4* macro. The mailer types that are known in this distribution are fax, local, smtp, uucp, and usenet. For example, to include support for the UUCP-based mailers, use ″MAILER(uucp)″.

ostype
: Definitions describing various operating system environments (such as the location of support files). These are referenced using the OSTYPE *m4* macro. This directory contains only the os390 definition.

README
: Contains all the latest information regarding this latest version of sendmail from the *www.sendmail.org* site

sh
: Shell files used by the *m4* build process.

siteconfig
: Local UUCP connectivity information. These normally contain lists of site information, for example:

  - SITE(contessa)
  - SITE(hoptoad)
  - SITE(nkainc)
  - SITE(well)

  These are referenced using the SITECONFIG macro:

  ```
  SITECONFIG(site.config.file, name_of_site,X)
  ```

  where *X* is the macro/class name to use. It can be U (indicating locally connected hosts) or one of W, X, or Y for up to three remote UUCP hubs.

This directory has been supplanted by the mailer table feature. Any new configurations should use that feature to do UUCP (and other) routing.

sendmail.ps
This is a PS file of the *Sendmail Installation and Operation Guide* provided by *www.sendmail.org* in this version of sendmail.

Creating the Configuration File

The basic steps to create the configuration file are:

1. Retrieve the m4 preprocessor.
2. Create the *.mc* file.
3. Build the configuration file.

Retrieving the m4 Preprocessor: Retrieve the m4 macro preprocessor from the z/OS Toys and Tools webpage at http://www.s390.ibm.com/unix/bpxa1toy.html.

The m4 macro preprocessor can be given input that will generate an z/OS UNIX sendmail configuration file. It takes as input a user-defined master configuration source file (.mc file) that can define mail delivery mechanisms using files provided in the samples directory. For more information on the .mc file, see “Creating the .mc File”.

The m4 preprocessor is downloaded as m4_pax.Z. To *unpax* the file, issue the following command:

```
pax -rzf m4_pax.Z
```

Creating the .mc File: The process of building an z/OS UNIX sendmail configuration file begins by creating a file of m4 statements. The suffix for this file is *.mc*.

The Minimal mc File: Every **.mc** file must contain minimal information. This file defines the mail delivery mechanisms understood at this site, how to access them, how to forward e-mail to remote mail systems, and a number of tuning parameters. The following table shows which items are required and also which items are recommended. It is recommended that the starting point for these items be as shown in the sample.mc file, and an investigation of all the m4 techniques that are available to customize the .mc file for your mail server is encouraged [refer to *sendmail* by O’Reilly & Associates, Inc. (ISBN 1-56592-222-0); this book is also referred to as the *bat book*].

Table 21. Required and Recommended m4 Items

Item	*Bat Book* **Reference**	**Required or Recommended**	**Description**
OSTYPE()	19.3.1	Required	Support for your operating system
MAILER()	19.3.2	Required	Necessary delivery agent
DOMAIN()	19.3.3	Recommended	Common domain wide information
FEATURE()	19.3.4	Recommended	Solutions to special needs

Example files can be found in the `/usr/lpp/tcpip/samples/sendmail` directory. The *cf* directory contains an example of an **.mc** file. Of special interest are the files that begin with *generic*. These can serve as template statements in developing customized **.mc** files. The following is an example of a simple **.mc** file.

```
divert (-1)
divert(0) dnl
VERSIONID('OS/390 sample configuration 12/4/97')
OSTYPE(os390)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

Following is a description of these common *m4* items. For more information on these items, refer to the *Bat Book.*

divert

- (-1) Ignore the lines following.
- (0) Stop diverting and output immediately.

VERSIONID

Used to insert an identifier into each **.mc** and **.m4** file that will become your header.

OSTYPE()

- Support for operating system (the only ostype provided in the `/usr/lpp/tcpip/samples/sendmail/ostype` directory is **os390.m4**).
- Required.

MAILER()

- Necessary delivery agent.
- Required.
- Known values include:
  - fax
  - local
  - smtp
  - uucp
  - usenet

DOMAIN()

Common domain wide information.

FEATURE()

Solution to special needs.

Building the Configuration File: To build the configuration file, go to the directory containing the m4 executable and issue the following command:

```
m4  ../m4/cf.m4 yourmcfile.mc > yourcffile.cf
```

where *yourmcfile* is the name of your .mc file and *yourcffile* is the name you want to give your .cf file.

The `../m4/cf.m4` specifies the master prototype configuration file `cf.m4` in the `m4` directory of the samples/sendmail directory. This is the path to the samples/sendmail directory structure from the location of your m4 executable. This can also be specified in your .mc file using *include* as follows:

```
include('../m4/cf.m4')
```

Creating the Aliases File

Aliasing is the process of converting one recipient name into another; a generic name (such as root) into a real username, or one name into a list of names (that is, a mailing list). Define the location of your aliases file using the AliasFile option in your sendmail.cf file. For example:

```
AliasFile=/etc/aliases
```

For sendmail to work, aliases are required for MAILER-DAEMON and postmaster. Every aliases file must include these required aliases.

The alias for postmaster must expand to the name of a real user, based on the requirement that every site has to be able to accept mail addressed to a user named postmaster. Unless a site has real user account named postmaster, an alias is required in the aliases file. The postmaster receives mail about mail problems sent by mail-related programs and by users that are having trouble sending mail.

When mail is bounced (returned because it could not be delivered), it is sent from MAILER-DAEMON but it is shown as being the original sender who sent the mail. This alias is defined because users often inadvertently reply to the bounced mail.

Following is an example of an aliases file. Lines that begin with # are comments. Empty lines are ignored. For more information on the different forms of aliases, refer to the *bat book.*

```
# Alias for mailer daemon
MAILER-DAEMON:IBMUSER

# Following alias is required by the new mail protocol, RFC 822
postmaster:IBMUSER

# Alias to handle mail to msgs and news
nobody: /dev/null
```

After the aliases file is created and before the sendmail daemon is brought up for the first time, the aliases file must be loaded by running sendmail using the *newaliases* command or with the *-bi* command-line switch.

For more information on the aliases file, refer to the *Bat Book.*

Configuration Hints and Tips

This section contains other required or useful information for configuring sendmail. For further information on these topics, refer to the *Bat Book.*

- SuperUser status is needed to start the sendmail daemon.
- The QueueDirectory option defined in the config file tells sendmail where to queue messages that are temporarily undeliverable. This directory must exist before sendmail is started.
- Sendmail is highly dependent on the Domain Name Server (DNS); it is important that the resolver be set up correctly to avoid unnecessary searching for a user. For more information on DNS, see “Chapter 8. Domain Name System (DNS)” on page 329.
- Table 22 on page 534 shows the expected file permissions of files that sendmail might use.

Table 22. Sendmail Permission Table

Path	Type	Owner	Mode		"Bat Book" Reference
/	Directory	root	0755	drwxr-xr-x	22.1
/usr	Directory	root	0755	drwxr-xr-x	18.8.34
/usr/sbin/sendmail	File	root	06511	-r-s--s--x	Entire Book
/etc	Directory	root	0755	drwxr-xr-x	18.8.34
/etc/sendmail.cf	File	root	0644 or 0640		Chapter 27
/etc/sendmail.st	File	root	0644	-rw-r--r--	26.6
/etc/sendmail.hf	File	root	0444	-r--r--r--	34.8.28
/etc/aliases	File	root	0644	-rw-r--r--	Chapter 24
/etc/aliases.pag	File	root	0644	-rw-r--r--	24.5
/etc/aliases.dir	File	root	0644	-rw-r--r--	24.5
/etc/aliases.db	File	root	0644	-rw-r--r--	24.5

If a system has thousands of users defined in the Users list, the administrator might consider enabling the UNIXMAP class. This increases the speed of the security checks performed by sendmail. APAR OW30858 provides details about what is needed to enable the UNIXMAP class.

For additional information about enabling the UNIX map class, refer to *z/OS SecureWay Security Server RACF Migration*.

Configuring Popper

Popper must be invoked by INETD upon the initiation of a TCP connection to the POP3 port 110 (or any other specifically-configured port defined in /etc/services - port 110 is the well-known port for POP3 protocols). Therefore, you must add the following command lines to your /etc/inetd.conf file:

```
pop3 stream tcp nowait bpxroot /usr/sbin/popper popper -d
```

The above must be added to your /etc/inetd.conf file and INETD must be started with this configuration file. For more information on inetd, see *z/OS Communications Server: IP Configuration Reference*.

POP3 resides on port 110. You can define additional ports if there is a need for additional command-line options for popper. For information on the options that might be suitable for your site, see the *z/OS Communications Server: IP User's Guide*.

z/OS UNIX popper will most likely be used by those whose local mailer requires a POP3 server. Typically their administrator will provide them with the address or name of the z/OS running the POP3 server, with instructions on where this information should be used.

Chapter 15. TIMED Daemon

TIMED is a TCP/IP daemon that is used to provide the time. TIMED gives the time in seconds since midnight January 1, 1900. You can start TIMED from the z/OS shell or as a started procedure. Each of these methods is described in *z/OS Communications Server: IP Configuration Reference*.

Starting TIMED from z/OS Shell

TIMED is installed in the /usr/lpp/tcpip/sbin/ directory.

To start the TIMED server from the command line, type the timed command.

```
timed [-l] [-p port]
```

Following are the parameters used for the timed command:

-l Logs all the incoming requests and responses to the system log. Logged information includes the IP address of the requestor.

-p port
Uses the specified port. The TIMED server usually receives requests on well-known port 37. You can specify the port in which requests are to be received.

Starting TIMED as a Procedure

The following sample shows how to start TIMED as a procedure.

```
//TIMED    PROC
//*
//* TCP/IP for MVS Network Station Manager Feature
//* SMP/E distribution name: EZATTMDP
//*
//*   5645-001 5655-HAL (C) Copyright IBM Corp. 1997.
//*   Licensed Materials - Property of IBM
//*   This product contains "Restricted Materials of IBM"
//*   All rights reserved.
//*   US Government Users Restricted Rights -
//*   Use, duplication or disclosure restricted by
//*   GSA ADP Schedule Contract with IBM Corp.
//*   See IBM Copyright Instructions.
//*
//* Function: Time server start procedure
//*
//TIMED    EXEC PGM=TIMED,REGION=0K,TIME=NOLIMIT,
//         PARM='POSIX(ON),ALL31(ON),TRAP(OFF)/'
//*STEPLIB DD DISP=SHR,DSN=TCP.SEZALINK,
//*           VOL=SER=,UNIT=
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//         PEND
```

Chapter 16. Remote Execution

This chapter describes how to configure and operate both the Remote Execution server and the UNIX Remote Execution server.

UNIX REXEC

The UNIX Remote Execution Protocol Daemon (REXECD) is the server for the REXEC routine. REXECD allows execution of z/OS UNIX commands with authentication based on user names and passwords.

The Remote Shell Server (RSHD) is the server for the remote shell (RSH) client. The server provides remote execution facilities with authentication based on privileged port numbers, user IDs, and passwords.

See “Configuring the z/OS UNIX Remote Execution Server” on page 540 for more information about configuring this server.

REXEC

The Remote Execution server allows execution of a TSO command that has been received at a remote host. This server runs the Remote EXecution Command Daemon (REXECD) which supports both the Remote Execution (REXEC) and Remote Shell (RSH) protocols.

This chapter describes how to configure and operate the Remote Execution server.

Configuring the Remote Execution Server

Steps to configure the Remote Execution server:

1. Update the Remote Execution cataloged procedure.
2. Update AUTOLOG and PORT statements in the PROFILE.TCPIP data set.
3. Determine whether the Remote Execution client will send a Remote Execution (REXEC) command or Remote Shell (RSH) command.
4. Permit remote users to access MVS resources. (Required only if the client is not sending a password.)
5. Create a user exit routine (optional).

Step 1: Configuring PROFILE.TCPIP for REXEC

If you want the Remote Execution server to start automatically when the TCPIP address space is started, include the name of the member containing the RXSERVE cataloged procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  RXSERVE
ENDAUTOLOG
```

To ensure that port 512 is reserved for the Remote Execution protocol and port 514 for the Remote Shell protocol, add the name of the member containing the Remote Execution cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  512 TCP RXSERVE
  514 TCP RXSERVE
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Determine Whether Remote Execution Client Will Send REXEC or RSH Commands

The Remote Execution client can send commands to the Remote Execution server by the following methods:

1. Sending the Remote Execution (REXEC) command
2. Sending the Remote Shell (RSH) command with a user ID and password separated by a slash (/) character with the -l option on the RSH command
3. Sending the Remote Shell (RSH) command without a password

With methods 1 and 2, the Remote Execution server executes the request and passes the password to MVS for verification. (REXEC commands require a password.) When these methods are used, skip Step 3.

With method 3, to enable an RSH client to send RSH commands to the MVS Remote Execution server without specifying a password, Step 3 is required.

Step 3: Permit Remote Users to Access MVS Resources (Optional)

This step is necessary only if your installation allows users to issue remote execution commands without the requirement of specifying a password on the remote execution client.

Use the following steps to ensure that the server can correctly access necessary MVS resources. You can use z/OS Security Server (RACF) or an equivalent security program.

1. Verify that your system has been configured for allowing surrogate job submission as described in *z/OS SecureWay Security Server RACF Security Administrator's Guide* (SC28-1915) or by using an equivalent security program.
2. Authorize the Remote Execution Server to submit jobs for the MVS user ID specified with the -l option of the RSH command. This can be done with the RACF facility as described in *z/OS SecureWay Security Server RACF General User's Guide* (SC28-1917), or by using an equivalent security program.
3. Define an *mvs_userid*.RHOSTS.DATA data set and authorize the Remote Execution Server userid permission to read this data set. This can be done with the RACF facility as described in *z/OS SecureWay Security Server RACF General User's Guide* (SC28-1917), or by using an equivalent security program.

   Note: This is the userid used to start the RXSERVE address space.

   This data set identifies the Remote Execution clients that can execute MVS commands remotely by sending an RSH command.

   When a Remote Execution client sends an RSH request to the Remote Execution server, the request includes the local user ID of the client user (*local_userID*) and, if the client user specified the -l option of the RSH command, the request also contains the user ID to use on the remote host (*mvs_userid*). If the client does not specify the -l option, the user ID to be used on the remote host is assumed to be the same as the *local_userID*.

   When the Remote Execution server receives an RSH command without a password, the server looks for a data set called *mvs_userid*.RHOSTS.DATA. If

the data set exists, the server reads it and looks for an entry with a host name that matches the client user's host. If the user ID specified on this entry in the RHOSTS.DATA data set matches the *local_userID* passed on the RSH command, the RSH command continues processing. If the entry does not exist, the server responds to the client with message `EZA4386E Permission denied`.

In the following example of an RHOSTS.DATA data set, the MVS client user `mvsuser` is allowed to issue the RSH command without a password from host `rs60007` with a local AIX user ID of `mvsuser`.

Example of mvsuser.RHOSTS.DATA data set:

```
rs60007.itso.ral.ibm.com mvsuser
```

Step 4: Update the Remote Execution Cataloged Procedure

Update the Remote Execution cataloged procedure by copying the sample provided in *hlq*.SEZAINST(RXPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify the Remote Execution server parameters and modify the JCL as required for your installation.

You can update the Remote Execution server operating parameters during execution with the MODIFY command. All but MAXCONN can be changed.

Step 5: Create a User Exit Routine (Optional)

Optionally, you can provide a user exit routine. This routine can be used to alter the JOB and EXEC statement parameters to meet installation-specific requirements such as which system should process the job and/or environment unique accounting information prior to submission of the TSO batch job.

The user exit should have the AMODE(31) attribute to provide addressability to the input parameter.

On entry to the user exit, register 1 points to the following parameter list:

Offset	Description
0	A pointer to a mixed INET address
4	A pointer to JOB statement parameters
8	A pointer to EXEC statement parameters
12	A pointer to an optional JES control statement

The INET address consists of the following fields:

Offset	Description
0	2 bytes (AF_INET or 2)
2	2 bytes (server port)
4	4 bytes (client INET address)

The JOB statement parameters can be up to 1024 characters in length and are ended by X'00'. You can modify the parameters with the exit routine. Upon entry, the parameters are set to:

- *user_ID*
- USER=*user_id*
- PASSWORD=*password*
- MSGCLASS=*msgclass*

MSGCLASS is as specified in the Remote Execution cataloged procedure. *userid* and PASSWORD are as received from the requesting client.

For RSH commands without passwords, note that the PASSWORD= parameter is not present. The *userid* in the first positional parameter can be processed by an installation-written JES exit.

The EXEC statement parameters can be up to 256 bytes in length and are ended by X'00'. These parameters can be modified by the exit routine. On entry, it contains the EXEC statement for the procedure specified in the TSOPROC parameter of the Remote Execution server or the default IKJACCNT procedure if TSOPROC is not specified.

The JES control statement parameter can be up to 256 bytes in length and is ended by X'00'. Upon entry, the parameter field is set to X'00'. Any JES control statement added by the user exit will be put between the JOB and the EXEC statement.

The modified JOB and EXEC statements are submitted as a TSO batch job.

The user exit is shipped as a sample in the RXUEXIT member of the SEZAINST data set. Refer to the REXEC chapter in *z/OS Communications Server: IP Configuration Reference* for more information about this sample.

Configuring the z/OS UNIX Remote Execution Server

Installation Information

This section describes the HFS files used by z/OS UNIX REXECD and RSHD.

HFS Files for z/OS UNIX REXECD

Note: The userid associated with the daemon in /etc/inetd.conf requires superuser authority. See ″Setting Up for Daemons″ in *z/OS UNIX System Services Planning* for a description of the kinds of authority defined for daemons.

The HFS files used by z/OS UNIX REXECD and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/orexecd
The server.

If BPX.DAEMON is specified, then the sticky bit must be set on, and /usr/sbin/orexecd, and orexecd can reside in an authorized MVS data set.

/usr/lib/nls/msg/C/rexdmsg.cat
The message catalog used by the z/OS UNIX REXECD server.

Note: This is not an actual member at this location, but it is a symbolic link to the part in /usr/lpp/tcpip/nls/msg/C/*.

Where the server looks for the message catalog (rexdmsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If rexdmsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

HFS Files for z/OS UNIX RSHD

The HFS files used by z/OS UNIX RSHD and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/orshd
The server.

If BPX.DAEMON is specified, the sticky bit must be set on, and /usr/sbin/orshd, and orshd can reside in an authorized MVS data set.

/usr/sbin/ruserok
An optional user exit that will authenticate users logging into the z/OS UNIX RSHD server with a null password. See “Setting up the z/OS UNIX RSHD Installation Exit” below for more information.

Note: This exit is required to allow support for null passwords with RSH.

/usr/lib/nls/msg/C/rshdmsg.cat
The message catalog associated with the z/OS UNIX RSHD client is stored here. If this file does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

Note: The message catalog is not actually stored here. This is a symbolic link, and the actual member is in /usr/lpp/tcpip/nls/msg/C/*.

Setting up the z/OS UNIX RSHD Installation Exit

When the -r option is enabled, if there is no password specified on the RSH command from the client, z/OS UNIX RSHD will drive the installation exit. When the installation exit is driven, RSHD looks for a program in /usr/sbin named ruserok. This is the only name that it will look for. If /usr/sbin/ruserok is not found, the request will fail.

When the z/OS UNIX RSHD server invokes /user/sbin/ruserok, it will pass parameters in the following order:

1. host name
2. local user’s UID
3. remote userid
4. local userid

If z/OS UNIX RSHD receives a return code of zero from the installation exit, z/OS UNIX RSHD continues. Any nonzero return code from the installation exit will cause RSHD to issue message EZYRS25E to the client and terminate all connections. The following code fragment can be used as an example to begin building a working ruserok installation exit:

```
int main(argc, argv)
     int argc;
     char *argv[];
  char *rhost1;    /* "hostname" or "hostname.domain" of client
                      obtained by caller:
                      gethostbyaddr(getpeername()) */
  int locuid;      /* uid of the user name on local system */
  char *cliuname;  /* user name on client's system */
  char *servuname; /* user name on this (server's) system */
  int rc = 4;

  rhost1 = argv[1];
  locuid = atoi(argv[2]);
  cliuname = argv[3];
  servuname = argv[4];
  .
  <authenticate user and set rc=0 if valid>
  .
  return(rc);
```

Chapter 17. Miscellaneous (MISC) Server

The Miscellaneous (MISC) server is a server that can be used to test and debug applications.

The MISC server supports the 3 protocols described in RFCs 862, 863, and 864:
- Discard
- Echo
- Character Generator

Discard Protocol

The MISC server simply throws away any data it receives. A TCP-based server listens for TCP connections on TCP port 9. If a connection is established, the data is discarded and no response is sent. A UDP-based server listens for UDP datagrams on UDP port 9. When a datagram is received, it is discarded and no response is sent.

Echo Protocol

The MISC server returns to the originating application any data that it receives. A TCP-based server listens for TCP connections on TCP port 7. Once a connection is established, any data that is received is sent back to the originating application. A UDP-based server listens for UDP datagrams on UDP port 7. When a datagram is received, the data it contained is sent back as an answering datagram.

Character Generator Protocol

The MISC server sends a repetitive stream of character data without regard to its content. A TCP-based server listens for TCP connections on TCP port 19. When a connection is established, a stream of data is sent to the connecting application. Any data that is received is thrown away. A UDP-based server listens for UDP datagrams on port 19. When a datagram is received, an answering datagram is sent that contains a random number (between 0 and 512) of characters. The data in the received datagram is ignored.

The data that is generated follows an ordered sequence. It repeats a pattern of 94 printable ASCII characters in a ring, so that character number 0 follows character number 94.

Following is an example of the repeated pattern.

```
!"#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefgh
"#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghi
#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghij
$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijk
%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijkl
&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklm
'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmn
()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmno
)*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnop
*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopq
└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqr,-
./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrs
-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrst
./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstu
/0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuv
0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvw
123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwx
23456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxy
3456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz
456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{
56789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|
6789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}
789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~
89:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~
9:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~ !
:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_'abcdefghijklmnopqrstuvwxyz{|}~ !"
```

Configuring the MISC Server

1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the MISC Server cataloged procedure (MISCSERV).

Step 1: Configuring PROFILE.TCPIP for the MISC Server

To allow the MISC server to start automatically when TCPIP is initialized, include the member name of the MISC server cataloged procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP.

```
AUTOLOG
  MISCSERV
ENDAUTOLOG
```

The AUTOLOG entry in *hlq*.PROFILE.TCPIP is optional. You can choose to start the MISC server manually, when it is needed, using the START command:

```
START MISCSERV
```

The MISC server requires ports 7, 9, and 19 for both TCP and UDP. To ensure that these ports are reserved for the MISC server, verify that they are assigned to the member containing the MISC server cataloged procedure in the PORT statement in PROFILE.TCPIP.

```
PORT
    7 UDP MISCSERV
    7 TCP MISCSERV
```

```
 9 UDP MISCSERV
 9 TCP MISCSERV
19 UDP MISCSERV
19 TCP MISCSERV
```

For more information on these statements, see the *z/OS Communications Server: IP Configuration Reference*.

Step 2: Updating the MISC Server Cataloged Procedure (MISCSERV)

Update the MISC server cataloged procedure by copying the sample in *hlq*.SEZAINST(MISCSERV) to your system or recognized PROCLIB and modifying the parameters and data set names to suit your local conditions.

MISC Server Cataloged Procedure (MISCSERV)

```
//MISCSERV PROC MODULE=MISCSRV,PARMS=''
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: SEZAINST(MISCSERV)
//*
//*        Licensed Materials - Program Property of IBM.
//*        This product contains "Restricted Materials of IBM"
//*        5685-061 (C) COPYRIGHT IBM CORP. 1994
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted
//*        by GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions
//*
//MISCSERV EXEC PGM=&MODULE,
//           REGION=4096K,TIME=1440,
//           PARM='&PARMS'
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here. If you add
//*        them to STEPLIB, they must be APF authorized.  Change
//*        the name as appropriate for your installation.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        MSMISCSR identifies an optional data set for NLS support.
//*        It specifies the MISC server message repository.
//*
//*MSMISCSR DD DSN=TCPIP.SEZAINST(MSMISCSR),DISP=SHR
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA.
//*        The SYSTCPD DD statement should be placed in the TSO logon
//*        procedure or in the JCL of any client or server executed
//*        as a background task.  The data set can be any sequential
//*        data set or a member of a partitioned data set (PDS).
//*
//*        For more information please see "Understanding TCP/IP Data
//*        Set Names" in the Customization and Administration Guide.
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Specifying the MISC Server Parameters

The MISC server generates periodic messages whenever a client sends data to ports 7, 9, or 19. If this server runs continually for a long period of time, considerable amounts of spool space can be consumed. Therefore, the MISC server has all tracing turned off by default.

You can enable the trace options for any of the three MISC server protocols using the PARMS= parameter on the PROC statement of the cataloged procedure. These options will be in effect when the server starts.

TRACE
Turns on tracing for any of the specified protocols and must be followed by one or more of these three keywords:

ECho Specifies tracing for the echo protocol on port 7

DIscard
Specifies tracing for the discard protocol on port 9

CHargen
Specifies tracing for the character generator protocol on port 19

DEbug
Specifies tracing for problem determination

For example, the following statement turns tracing on for the echo and discard protocols.

```
//MISCSERV PROC MODULE=MISCSRV,PARMS='TRACE ECHO DISCARD'
```

Part 3. Appendixes

Appendix A. Setting up the inetd Configuration File

inetd is a generic listener program used by such servers as z/OS UNIX telnet server and z/OS UNIX rexec server. Other servers such as z/OS UNIX ftp server have their own listener program and do not use inetd.

inetd.conf is an example of the user's configuration file. It is stored in the /etc directory. Ensure that the inetd services required on your system are enabled using configuration statements like those in the following example:

```
#=====================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
 shell    stream    tcp        nowait  OMVSKERN /usr/sbin/orshd rshd -l
 exec     stream    tcp        nowait  OMVSKERN /usr/sbin/orexecd rexecd -LV
 otelnet  stream    tcp        nowait  OMVSKERN /usr/sbin/otelnetd otelnetd  -LV
```

Figure 63. Adding Applications to /etc/inetd.conf

To establish a relationship between the servers defined in the /etc/inetd.conf file and specific port numbers in the OpenEdition environment, insure that statements have been added to ETC.SERVICES for each of these servers. See the sample ETC.SERVICES installed in the /usr/lpp/tcpip/samples/services directory for how to specify ETC.SERVICES statements for these servers.

The traces for both the OE REXECD server and the OE RSHD server are enabled via options in the inetd configuration file (/etc/inetd.conf):
The traces are turned on for both servers by passing a -d argument to the server

```
#=====================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
shell      stream    tcp       nowait OMVSKERN /usr/sbin/orshd rshd -d  1
exec       stream    tcp       nowait OMVSKERN /usr/sbin/orexecd rexecd -d  2
```

Figure 64. Setting Traces in /etc/inetd.conf

programs. 1 is the RSHD server and 2 is the REXECD server. All commands executed after the debug flags have been turned on in the inetd configuration file and the inetd server has reread the file will produce trace output.

The trace is written in formatted form to the syslogd facility name daemon with a priority of debug. The trace data can be routed to a file in your Hierarchical File System by specifying the following definition in your syslogd configuration file (/etc/syslogd.conf):

```
  #
  # All ftp, rexecd, rshd
  # debug messages (and above
  # priority messages) go
# to server.debug.a
```

```
#
daemon.debug              /tmp/syslogd/server.debug.a
```

In this example, the trace data is written to /tmp/syslogd/daemon.debug.a in your Hierarchical File System. For more information on syslogd, refer to “Logging of System Messages” on page 25.

For more information about inetd, refer to *z/OS UNIX System Services Planning*, GA22-7800 or *Accessing OS/390 OpenEdition MVS from the Internet* (SG24-4721).

Appendix B. Related Protocol Specifications

This appendix lists the related protocol specifications for TCP/IP for MVS. The Internet suite of protocols 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 recommended protocols. 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 MVS are based on the following RFCs:

RFC	Title and Author
768	*User Datagram Protocol* J.B. Postel
791	*Internet Protocol* J.B. Postel
792	*Internet Control Message Protocol* J.B. Postel
793	*Transmission Control Protocol* J.B. Postel
821	*Simple Mail Transfer Protocol* J.B. Postel
822	*Standard for the Format of ARPA Internet Text Messages* D. Crocker
823	*DARPA Internet Gateway* R.M. Hinden, A. Sheltzer
826	*Ethernet Address Resolution Protocol: or Converting Network Protocol Addresses to 48.Bit Ethernet Address for Transmission on Ethernet Hardware* D.C. Plummer
854	*Telnet Protocol Specification* J.B. Postel, J.K. Reynolds
855	*Telnet Option Specification* J.B. Postel, J.K. Reynolds
856	*Telnet Binary Transmission* J.B. Postel, J.K. Reynolds
857	*Telnet Echo Option* J.B. Postel, J.K. Reynolds
858	*Telnet Suppress Go Ahead Option* J.B. Postel, J.K. Reynolds
859	*Telnet Status Option* J.B. Postel, J.K. Reynolds
860	*Telnet Timing Mark Option* J.B. Postel, J.K. Reynolds
861	*Telnet Extended Options —List Option* J.B. Postel, J.K. Reynolds
862	*Echo Protocol* J.B. Postel
863	*Discard Protocol* J.B. Postel
864	*Character Generator Protocol* J.B. Postel
877	*Standard for the Transmission of IP Datagrams over Public Data Networks* J.T. Korb
885	*Telnet End of Record Option* J.B. Postel
903	*Reverse Address Resolution Protocol* R. Finlayson, T. Mann, J.C. Mogul, M. Theimer
904	*Exterior Gateway Protocol Formal Specification* D.L. Mills
919	*Broadcasting Internet Datagrams* J.C. Mogul
922	*Broadcasting Internet Datagrams in the Presence of Subnets* J.C. Mogul

950 *Internet Standard Subnetting Procedure* J.C. Mogul, J.B. Postel

952 *DoD Internet Host Table Specification* K. Harrenstien, M.K. Stahl, E.J. Feinler

959 *File Transfer Protocol* J.B. Postel, J.K. Reynolds

974 *Mail Routing and the Domain Name System* C. Partridge

1009 *Requirements for Internet Gateways* R.T. Braden, J.B. Postel

1013 *X Window System Protocol, Version 11: Alpha Update* R.W. Scheifler

1014 *XDR: External Data Representation Standard* Sun Microsystems Incorporated

1027 *Using ARP to Implement Transparent Subnet Gateways* S. Carl-Mitchell, J.S. Quarterman

1032 *Domain Administrators Guide* M.K. Stahl

1033 *Domain Administrators Operations Guide* M. Lottor

1034 *Domain Names—Concepts and Facilities* P.V. Mockapetris

1035 *Domain Names—Implementation and Specification* P.V. Mockapetris

1042 *Standard for the Transmission of IP Datagrams over IEEE 802 Networks* J.B. Postel, J.K. Reynolds

1044 *Internet Protocol on Network System's HYPERchannel: Protocol Specification* K. Hardwick, J. Lekashman

1055 *Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP* J.L. Romkey

1057 *RPC: Remote Procedure Call Protocol Version 2 Specification* Sun Microsystems Incorporated

1058 *Routing Information Protocol* C.L. Hedrick

1073 *Telnet Window Size Option* D. Waitzman

1079 *Telnet Terminal Speed Option* C.L. Hedrick

1091 *Telnet Terminal-Type Option* J. VanBokkelen

1094 *NFS: Network File System Protocol Specification* Sun Microsystems Incorporated

1096 *Telnet X Display Location Option* G. Marcy

1112 *Host Extensions for IP Multicasting* S. Deering

1118 *Hitchhikers Guide to the Internet* E. Krol

1122 *Requirements for Internet Hosts—Communication Layers* R.T. Braden

1123 *Requirements for Internet Hosts—Application and Support* R.T. Braden

1155 *Structure and Identification of Management Information for TCP/IP-Based Internets* M.T. Rose, K. McCloghrie

1156 *Management Information Base for Network Management of TCP/IP-based Internets* K. McCloghrie, M.T. Rose

1157 *Simple Network Management Protocol (SNMP)* J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin

1179 *Line Printer Daemon Protocol* The Wollongong Group, L. McLaughlin III

1180 *TCP/IP Tutorial* T.J. Socolofsky, C.J. Kale

1183 *New DNS RR Definitions* C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris, (Updates RFC 1034, RFC 1035)

1184 *Telnet Linemode Option* D. Borman

1187 *Bulk Table Retrieval with the SNMP* M.T. Rose, K. McCloghrie, J.R. Davin

1188 *Proposed Standard for the Transmission of IP Datagrams over FDDI Networks* D. Katz

1191 *Path MTU Discovery* J. Mogul, S. Deering

1198 *FYI on the X Window System* R.W. Scheifler

1207 *FYI on Questions and Answers: Answers to Commonly Asked Experienced Internet User Questions* G.S. Malkin, A.N. Marine, J.K. Reynolds

1208 *Glossary of Networking Terms* O.J. Jacobsen, D.C. Lynch

1213 *Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II,* K. McCloghrie, M.T. Rose

1215 *Convention for Defining Traps for Use with the SNMP* M.T. Rose

1228 *SNMP-DPI Simple Network Management Protocol Distributed Program Interface* G.C. Carpenter, B. Wijnen

1229 *Extensions to the Generic-Interface MIB* K. McCloghrie

1230 *IEEE 802.4 Token Bus MIB IEEE 802 4 Token Bus MIB* K. McCloghrie, R. Fox

1231 *IEEE 802.5 Token Ring MIB IEEE 802.5 Token Ring MIB* K. McCloghrie, R. Fox, E. Decker

1267 *A Border Gateway Protocol 3 (BGP-3)* K. Lougheed, Y. Rekhter

1268 *Application of the Border Gateway Protocol in the Internet* Y. Rekhter, P. Gross

1269 *Definitions of Managed Objects for the Border Gateway Protocol (Version 3)* S. Willis, J. Burruss

1270 *SNMP Communications Services* F. Kastenholz, ed.

1323 *TCP Extensions for High Performance* V. Jacobson, R. Braden, D. Borman

1340 *Assigned Numbers* J.K. Reynolds, J.B. Postel

1348 *DNS NSAP RRs* B. Manning

1350 *TFTP Protocol* K.R. Sollins

1351 *SNMP Administrative Model* J. Davin, J. Galvin, K. McCloghrie

1352 *SNMP Security Protocols* J. Galvin, K. McCloghrie, J. Davin

1353 *Definitions of Managed Objects for Administration of SNMP Parties* K. McCloghrie, J. Davin, J. Galvin

1354 *IP Forwarding Table MIB* F. Baker

1356 *Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode* A. Malis, D. Robinson, R. Ullmann

1372 *Telnet Remote Flow Control Option* D. Borman, C. L. Hedrick

1381 *SNMP MIB Extension for X.25 LAPB* D. Throop, F. Baker

1382 *SNMP MIB Extension for the X.25 Packet Layer* D. Throop

1390 *Transmission of IP and ARP over FDDI Networks* D. Katz

1393 *Traceroute Using an IP Option* G. Malkin

1397 *Default Route Advertisement In BGP2 And BGP3 Versions of the Border Gateway Protocol* D. Haskin

1398 *Definitions of Managed Objects for the Ethernet-like Interface Types* F. Kastenholz

1540 *IAB Official Protocol Standards* J.B. Postel

1571 *Telnet Environment Option Interoperability Issues* D. Borman

1572 *Telnet Environment Option* S. Alexander

1577 *Classical IP and ARP over ATM* M. Laubach

1592 *Simple Network Management Protocol Distributed Protocol Interface Version 2.0* B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters

1594 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* A.N. Marine, J. Reynolds, G.S. Malkin

1695 *Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2*M. Ahmed, K. Tesink

1723 *RIP Version 2 — Carrying Additional Information* G. Malkin

1850 *OSPF Version 2 Management Information Base* F. Baker, R. Coltun

1901 *Introduction to Community-Based SNMPv2* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1902 *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1903 *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1904 *Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1905 *Protocols Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1906 *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1907 *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1908 *Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

2011 *SNMPv2 Management Information Base for the Internet Protocol using SMIv2* K. McCloghrie

2012 *SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2* K. McCloghrie

2013 *SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2* K. McCloghrie

2233 *The Interfaces Group MIB using SMIv2* K. McCloghrie, F. Kastenholz

2271 *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

2272 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

2273 *SNMP Applications* David B. Levi, Paul Meyer, Bob Stewart

2274 *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

2275 *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

2320 *Definitions of Managed Objects for Classical IP and ARP Over ATM Using SMIv2* M. Greene, J. Luciani, K. White, T. Kuo

2358 *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson

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`.

Appendix C. Configuring the OROUTED Server

Before You Configure...

Read and understand “Chapter 1. Configuration Overview” on page 3. It covers important information about data set naming and search sequences.

This appendix describes how to configure the OROUTED server. It explains OROUTED’s use of the Routing Information Protocol to help you decide if this server is suitable for your network.

Notes:

1. OMPROUTE, which was introduced in eNetwork Communications Server for OS/390 V2R6 and enhanced in V2R7, is the recommended routing daemon. OROUTED will eventually be removed; you will receive ample formal notice of this change.
2. OROUTED does not support zero subnets.
3. OROUTED does not support equal-cost multipath routes to a destination network or host. However, OROUTED can be used in conjunction with statically defined equal-cost multipath routes in the GATEWAY statement of the TCP/IP profile.

Understanding OROUTED

The route daemon is a server that implements the Routing Information Protocols (RIP) described in RFC 1058 (RIP version 1) and in RFC 1723 (RIP version 2). It provides an alternative to the static TCP/IP gateway definitions. When configured properly the MVS host running with OROUTED becomes an active RIP router in a TCP/IP network. The OROUTED server dynamically creates and maintainsthe network routing tables using RIP. RIP allows gateways and routers to periodically broadcast their routing tables to adjacent nodes. This enables the OROUTED server to update the host routing table. For example, the OROUTED server can determine if a new route has been created, if a route is temporarily unavailable, or if a more efficient route exists. OROUTED has the following characteristics:-

- Deletion of all RIP routes at startup. The -del start parameter might be used to delete all dynamic routes from the routing table upon initialization of OROUTED.
- OROUTED is an z/OS UNIX application. It requires the Hierarchical File System (HFS) to run.
- OROUTED can be started from an MVS procedure or from the z/OS shell command line.
- OROUTED uses a standard message catalog. The message catalog must be in the HFS. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.
- All messages and trace information is sent to the syslogd, except for output from the *-d* and *-dp* parameters, which is sent to STDOUT.
- A default mode of operation is for the program to close STDIN, STDOUT and STDERR. This allows for userids to cleanly exit the z/OS shell after starting the program in the background. As a consequence, the program printf statements are also disabled. The parameter ″-ep″ enables printf’s. If this parameter is specified, the program should not be run in the background, because the userid will not be able to exit the shell until the background job has been killed.
- OROUTED and OMPROUTE cannot run on the same stack concurrently.

- OROUTED needs to be started by a RACF-authorized user ID.

Routing Information Protocol (RIP)

The Routing Information Protocol (RIP) is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. IGPs are used to manage the routing information of a single autonomous system, or a single piece of the TCP/IP network. RIP has many limitations and is not suited for every TCP/IP environment. Before installing the OROUTED server, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. See “Appendix B. Related Protocol Specifications” on page 551 for more information about RFC 1058 and RFC 1723.

RIP uses the number of hops, or *hop count*, to determine the best possible route to a host or network. The term *hop count* is also referred to as the *metric*. A gateway is defined as zero hops from directly connected networks, one hop from networks that can be reached through one gateway, and so on. In RIP, a hop count of 16 means infinity, or that the destination cannot be reached. This limits the longest path in the network that can be managed by RIP to 15 gateways.

The OROUTED server propagates routing information to the neighboring gateway's on gateway's directly connected networks every 30 seconds. The server receives updates from neighboring gateways periodically and uses this information to update the routing tables. If an update has not been received from a gateway in 180 seconds (3 minutes), OROUTED assumes the gateway is down and sets all the routes through that gateway to a metric of 16 (infinity). If an update has not been received from a gateway in another 120 seconds (2 minutes), OROUTED deletes all of the routes through that gateway.

During the intervals specified by the *interface.scan.interval* and *interface.poll.interval* values on the OPTIONS statement, OROUTED checks to determine if a local interface is up or down by scanning the TCP/IP interface tables. It also checks to see if an interface has been added or reactivated.

For networks that are not point-to-point, such as Token-Ring and Ethernet, OROUTED receives its own copy of broadcasted or multicasted RIP packets over the interfaces, provided that the interfaces are active. Other networks, such as point-to-point, can be managed by OROUTED as long as there are RIP services managing the other end of point-to-point link(s). If the destination addresses are defined for these point-to-point networks, the RIP packets are unicasted. Otherwise, the RIP packets are broadcasted or multicasted. In networks where there are adjacent routers or hosts not running RIP services, OROUTED will not be receiving updates over the links and eventually will delete all of the routes to these networks. To prevent the routes to these networks not running RIP services from being deleted, passive routes may be coded in a OROUTED gateways data set. For more information, see “OROUTED Gateways” on page 561 and “OROUTED Parameters” on page 574.

RIP Version 2

RIP Version 2 is an extension of RIP Version 1 and provides the following features:

Route Tags to provide EGP-RIP and BGP-RIP interactions
The route tags are used to separate ″internal″ RIP routes (routes for networks within the RIP routing domain) from ″external″ RIP routes, which

may have been imported from an EGP or another IGP. OROUTED will not generate route tags, but will preserve them in received routes and readvertise them when necessary.

Variable subnetting support
Variable-length subnet masks are being included in routing information so that dynamically-added routes to destinations outside subnetworks or networks can be reachable.

Immediate Next Hop for shorter paths
Next hop IP addresses, whenever applicable, are being included in the routing information. Its purpose is to eliminate packets being routed through extra hops in the network. OROUTED will not generate immediate next hops, but will preserve them if they are included in the RIP packets.

Multicasting for RIPv2 packets to reduce load on hosts
An IP multicast address 224.0.0.9, reserved for RIPv2 packets, is used to reduce unnecessary load on hosts which are not listening to RIPv2 messages. RIPv2 multicasting is dependent on interfaces that are multicast-capable. By default, RIPv1 packets will be broadcast for interfaces that are not multicast-capable.

Authentication of RIPv2 packets for routing update security
Authentication keys, consisting of passwords, can be configured to be included in the outgoing RIPv2 packets for authentication by adjacent routers as a routing update security protection. Likewise, incoming RIPv2 packets are checked against local authentication keys. Any outgoing or incoming RIPv1 packets are not authenticated. For maximum security, configure OROUTED such that it will supply and receive RIPv2 packets only in addition to specification of authentication keys. The authentication keys are configurable on a router-wide or per-interface basis.

Configuration switches for RIPv1 and RIPv2 packets
Configuration switches are provided to selectively control which versions of RIP packets are to be sent or received over network interfaces. The switches should be configured based upon the routing capabilities of the network and are configurable on a router-wide or per-interface basis.

Supernetting support
The supernetting feature is part of Classless InterDomain Routing (CIDR) function. Supernetting provides a way to combine multiple network routes into fewer 'supernet' routes. This means that the number of network routes in the routing tables becomes smaller for advertisements. Supernet routes are received and sent in RIPv2 messages. If local supernet routes are defined for OROUTED, they will be advertised to adjacent routers. Local supernet routes are generated by OROUTED for interfaces with subnet masks that are less than the network class mask in value.

OROUTED Miscellaneous Features

OROUTED supports the following miscellaneous features:

- Multiple Network Attachments.

  Multiple attachments and IP addresses on the same network are supported, providing redundant paths to other hosts or routers on directly-attached networks.
- Virtual IP Addressing (VIPA).

  If virtual IP addresses are configured on the z/OS server and there are multiple network attachments, OROUTED can be used to advertise the VIPA routes to the network, providing the necessary routing information to the adjacent routers or

hosts running RIP services. Using the VIPA routes, the adjacent routers or hosts will be able to reach the VIPA addresses as the destinations and to route around failures for fault tolerance support. For more information, see “Chapter 3. Virtual IP Addressing” on page 109.

RIP Input/Output Filters

The RIP input/output filters provide routing table manipulation and routing control. The filters are provided by OROUTED and consist of:

1. Route Blocking (or NoReceiving)
2. Route Forwarding (Unconditional and Conditional)
3. Route Receiving (Unconditional and Conditional)
4. Route NoForwarding
5. Interface Supply Switch
6. Interface RIP On/Off Switch
7. Default Route Only Supply Switch
8. Virtual Route Only Supply Switch
9. Default and Virtual Routes Only Supply Switch
10. Local (directly-connected) Routes Only Supply Switch
11. Triggered Updates Only Supply Switch
12. Gateway NoReceiving

For more information on these RIP input/output filters, see the OROUTED procedure parameters in “OROUTED Parameters” on page 574 and the options statement in “Step 6: Configure the Gateways File or Data Set (Optional)” on page 567.

RIP Routes

Dynamic routes are any routes created by the following dynamic routing applications:

- OMPROUTE (OSPF/RIP)
- OROUTED (RIP)
- NCPROUTE (RIP)
- ICMP Redirects
- Other (created by customer application)

However, RIP routes are dynamic routes created only by RIP applications, like OROUTED and OMPROUTE.

To help maintain the integrity of the routing table, at initialization, OROUTED attempts to delete all RIP routes from the stack routing table and then adds RIP routes based on BSDROUTINGPARMS and the optional gateways file. The -del start parameter can be used to delete all dynamic routes from the routing table upon initialization of OROUTED.

When OROUTED terminates, RIP routes are not deleted. If you want to remove all RIP routes upon OROUTED termination, used the -kdr MODIFY option.

OROUTED Gateways

Passive RIP Routes

Passive RIP routes are known by both TCP/IP and OROUTED. Information about passive routes is put in TCP/IP's and OROUTED's routing tables. A passive entry in OROUTED's routing table is used as a placeholder to prevent a route from being propagated and from being overwritten by a competing RIP route. With the exception of directly-connected passive routes, passive routes are not propagated; they are known only by this router. Using passive routes can create routing loops, so they need to be created carefully.

Defining passive routes such as these should be avoided:

A to C is via B.
B to C is via A.

Passive routes should be used when adding routes where the host/net is not running RIP. Passive routes should also be used when adding a default route, since this is the only way to prevent a route from timing out.

External RIP Routes

External RIP routes are known by OROUTED, but not by TCP/IP. External routes, such as the External Gateway Protocol (EGP), are managed by other protocols. The OROUTED server needs to know not to interfere with these and not to delete them.

An external entry exists in OROUTED's routing tables as a place holder to prevent a route from being overwritten by a competing RIP route. External routes are not propagated. OROUTED does not manage external routes. Therefore, OROUTED only knows that there is an existing route to host/net and one that is known to TCP/IP.

External routes should be used when the local host is running with some type of non-RIP routing protocol which dynamically changes the TCP/IP routing tables. The foreign host does not need to run any routing protocol, since the only concern is how to route traffic from the local host to the foreign host, and how to prevent multiple routing protocols from interfering with each other.

Active Gateways

Active gateways are treated as remote network interfaces. Active gateways are routers which are running RIP, but are reached by a medium which does not allow broadcasting or multicasting and is not point-to-point. OROUTED normally requires that routers be reachable via broadcast for non-point-to-point links or via unicast addresses for point-to-point links. If the interface is neither, then an active gateway entry can add the gateway to OROUTED's interface list. OROUTED will treat the active gateway as a remote network interface. Note that the active gateway must be directly connected.

Active gateways should be used when the foreign router is reachable over a non-broadcast and non-point-to-point network, and is directly connected to the local host.

OROUTED will communicate with active routers by unicast transmissions to the gateway address. Routes are not added to either OROUTED or the TCP/IP routing table immediately. They are added and propagated normally when route

advertisements arrive from an active gateway. The sole effect of an active gateway statement is to bypass the requirement for broadcast or multicast communication on non-point-to-point links. Interfaces which are not broadcast-capable or multicast-capable, not point-to-point, and are not active gateways are assumed to be loopback interfaces to the local host. Also, while a route to an active gateway might time out, the interface entry is never removed. If transmissions resume, then the new routes will still be available to the active gateways.

OROUTED Gateways Summary

Table 23 provides a summary of the OROUTED gateways and their characteristics.

Table 23. OROUTED Gateways Summary

	Propagated?	Known by TCP/IP?	Known by OROUTED?	Timeout?
Dynamic (1)	Yes	Yes	Yes	Yes
Passive	No (2)	Yes	Yes	No
External	No	No	Yes	No
Active	Yes	Yes	Yes	Yes
Dynamic routing provided by OROUTED.				

Notes:

1. Except directly-connected passive routes.
2. Directly-connected passive routes are propagated to other network interfaces for network reachability. A directly-connected passive route is one where the gateway address is one of the local interfaces in the HOME list.

OROUTED Configuration Process

The steps to configure OROUTED are as follows:

1. Configure the OROUTED profile.
2. Update PORT, BSDROUTINGPARMS, GATEWAY, BEGINROUTES, and IPCONFIG statements in the TCPIP profile.
3. Update the resolver configuration file.
4. Update the OROUTED cataloged procedure (optional).
5. Specify the OROUTED port number in the SERVICES file or data set.
6. Configure the gateways file or data set (optional).

   Note: If a default route is to be defined to a destination gateway or router, configure a default route in this gateways file or data set.
7. If not already started, configure and start syslogd.
8. RACF-authorize user IDs for starting OROUTED.

Step 1: Configure the OROUTED Profile

OROUTED supports sending and receiving both RIP version 1 and RIP version 2 packets. A configuration file, the OROUTED profile, determines the mode of operation. The following is the search order used to locate the OROUTED configuration data set or file:

1. If the environment variable ROUTED_PROFILE has been defined, OROUTED uses this value as the name of an MVS data set *(//'mvs.dataset.name')* or HFS file *(/dir/subdir/file.name)* to access the profile.

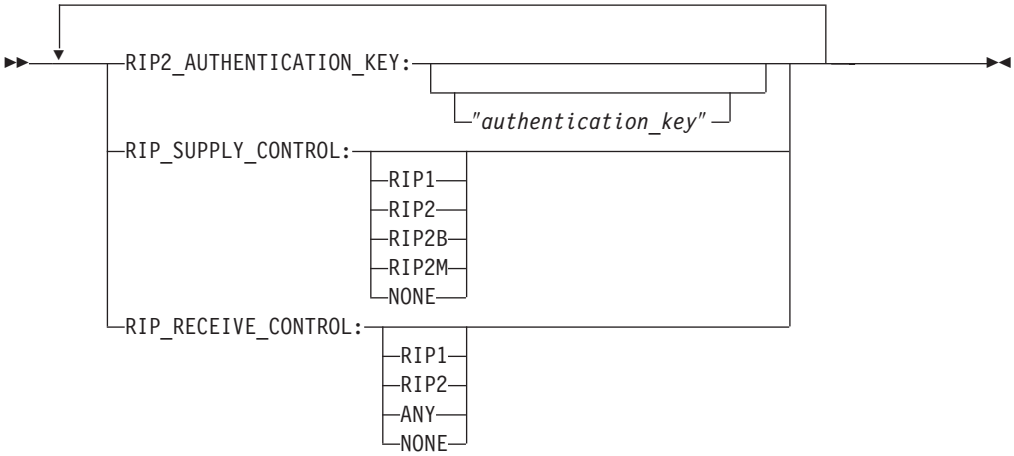
2. /etc/routed.profile
3. *hlq*.ROUTED.PROFILE

The following are the syntax rules for the OROUTED profile:

- Keywords can be specified in mixed case.
- Blanks and comments are supported in the OROUTED profile. Comments are identified by a semicolon in any column.
- Profile statements may start in any column; however, wrapping to the next record for continuation is not allowed.
- There should be no sequence numbers in the data set or file.

A sample profile is provided in *hlq*.SEZAINST(EZARTPRF).

The following are the options that can be included in the OROUTED profile:



RIP2_AUTHENTICATION_KEY:
A constant. The value that follows is the authentication key.

authentication key
Specifies a plain text password containing up to 16 characters. The authentication key must be enclosed in double quotes. The key is used on a server-wide basis and can contain mixed case and blank characters. The key will be used to authenticate RIP Version 2 packets and be included in the RIP responses for authentication by adjacent routers running RIP Version 2. A null key (either no key is specified, or "" is specified) indicates that authentication is disabled. For maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL to RIP2. This will discard RIP1 and unauthenticated RIP2 packets.

RIP_SUPPLY_CONTROL:
A constant. Specifies that the keyword following is to be used as the RIP supply control for all interfaces. Possible supply controls are as follows:

RIP1 Unicast/Broadcast RIP Version 1 packets (Default)

RIP2 Unicast/Multicast RIP Version 2 packets

RIP2B Unicast/Broadcast RIP Version 2 packets (Not Recommended)

RIP2M
Unicast/Multicast/Broadcast RIP packets (Migration)

NONE Disable sending RIP packets

RIP_RECEIVE_CONTROL:
A constant. Specifies that the keyword following is to be used as the RIP receive control for all interfaces. Possible receive controls are as follows:

RIP1 Receive RIP Version 1 packets

RIP2 Receive RIP Version 2 packets

ANY Receive any RIP Version 1 and 2 packets (Default)

NONE Disable receiving RIP packets

Figure 65 on page 565 is a sample OROUTED configuration file:

```
; EZARTPRF
;
; COPYRIGHT = NONE.
;
; Sample OROUTED profile.
; See the "IP Configuration Guide for more information"
;
; ---------------------------------------------------------------------
; RIP_SUPPLY_CONTROL specifies one of the following options on a
; router-wide basis:
;
; 1) RIP1  - Unicast/Broadcast RIP Version 1 packets (Default)
; 2) RIP2B - Unicast/Broadcast RIP Version 2 packets (Not Recommended)
; 3) RIP2M - Unicast/Multicast/Broadcast RIP packets (Migration)
; 4) RIP2  - Unicast/Multicast RIP Version 2 packets
; 5) NONE  - Disables sending RIP packets
;
; Note: If RIP2 is specified, the RIP Version 2 packets are multicast
;       over multicast-capable interfaces only.  No RIP packets are
;       sent over multicast-incapable interfaces.  For RIP2M, the RIP
;       Version 2 packets are multicast over multicast-capable
;       interfaces and RIP Version 1 packets are broadcast or unicast over
;       multicast-incapable interfaces.  For RIP2B, the RIP Version
;       2 packets are broadcast or unicast; this option is not recommended since
;       host route misinterpretations by adjacent routers running RIP
;       Version 1 can occur.  For this reason, RIP2B may become
;       obsolete in a future release.  For point-to-point interfaces
;       that are non-broadcast and multicast-incapable, the RIP
;       Version 2 packets are unicast.

RIP_SUPPLY_CONTROL: RIP1

; RIP_RECEIVE_CONTROL specifies one of the following options on a
; router-wide basis:
;
; 1) RIP1   - Receive RIP Version 1 packets only
; 2) RIP2   - Receive RIP Version 2 packets only
; 3) ANY    - Receive any RIP Version 1 and 2 packets (Default)
; 4) NONE   - Disables receiving RIP packets

RIP_RECEIVE_CONTROL: ANY

; RIP2_AUTHENTICATION_KEY specifies a plain text password containing
; up to 16 characters.  The authentication key must be enclosed in
; double quotes.  The key is used on a server-wide basis and can
; contain mixed case and blank characters.  The key will be used to
; authenticate RIP Version 2 packets and be included in the
; RIP responses for authentication by adjacent routers running RIP
; Version 2.  A null key (either no key is specified or "" is
; specified) indicates that authentication is disabled.  For
; maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL
; to RIP2.  This will discard RIP1 and unauthenticated RIP2 packets.

RIP2_AUTHENTICATION_KEY:
```

Figure 65. Sample OROUTED Configuration File

Step 2: Update Configuration Statements in PROFILE.TCPIP

To ensure that UDP port 520 is reserved for OROUTED, also add the name of the member containing the OROUTED cataloged procedure to the PORT statement in PROFILE.TCPIP:

```
PORT
   520 UDP OROUTED
```

In addition, configure the BSDROUTINGPARMS statements with your routing information. The BEGINROUTES or GATEWAY statement is not used and should be removed or commented from PROFILE.TCPIP.

If a static route must be coded in the GATEWAY statement in the TCP/IP Profile, a corresponding external route must be coded in ORouteD's /etc/gateways. This will inform ORouteD that these routes are locally defined and are not to be changed; in addition, by the external definition for a place holder in ORouteD's routing table, ORouteD will not attempt to delete or add equivalent routes from the RIP updates.

For multipath routes with common destination addresses reachable over multiple interfaces and/or multiple gateways or nexthops, code a corresponding external route from one of the multipath routes in /etc/gateways. Because ORouteD supports only one route to a destination and because the nexthop or gateway address is irrelevant in external route definitions, ORouteD will use the external route definition to represent any static route sharing a common destination address and prevents static route overrides from competing RIP routes.

Code the following on the IPCONFIG statement in PROFILE.TCPIP:

```
IPCONFIG IGNOREREDIRECT DATAGRAMFWD
```

Do not specify the no forwarding (NOFWD) option. To enable variable subnetting, add the VARSUBNETTING option. If OROUTED is to be used to either send or receive RIP version 2 packets, the VARSUBNETTING option must be specified in the TCPIP profile. To enable outbound source VIPA, add the SOURCEVIPA option.

Refer to *z/OS Communications Server: IP Configuration Reference* for descriptions and examples of these statements.

Note: If you want to be able to start OROUTED from the z/OS shell, use the special name OMVS as follows:

```
PORT 520 UDP OMVS
```

This enables the entire "OMVS job group" (that is, all z/OS shell users). Only RACF-authorized users can start OROUTED.

Step 3: Update the Resolver Configuration File

The resolver configuration file or data set contains keywords that are used by OROUTED. Two important keywords in the resolver file are DATASETPREFIX and TCPIPjobname. The value assigned to DATASETPREFIX will determine the high-level qualifier (*hlq*). The *hlq* is then used in the search order for other configuration files. If no DATASETPREFIX keyword is found in the resolver configuration data set or file, a default of TCPIP is used. In a CINET environment, the value assigned to TCPIPjobname will be used as the name of the stack with which OROUTED attempts to establish a connection. In an INET environment, it is not necessary to set TCPIPjobname, but if it is set, it must be set to ″INET″.

For a description of the search order used by the resolver to locate the resolver configuration data set or file, see "Resolver Configuration Files" on page 15.

Step 4: Update the OROUTED Cataloged Procedure (Optional)

If OROUTED is to be started by a procedure, update the cataloged procedure OROUTED by copying the sample in *hlq*.SEZAINST(OROUTED) to your system or recognized PROCLIB. Specify OROUTED parameters and change the data set names as required to suit your local configuration.

Note: When using PGM=BPXBATCH to start OROUTED, STDOUT and STDERR cannot be directed to any SYSOUT class; they must be directed to an HFS file.

OROUTED Cataloged Procedure

A data set or file may be used to set the environment variables for an invocation of OROUTED. This file is specified on the STDENV statement. An example of its contents is included in the OROUTED cataloged procedure sample. For more information about the cataloged procedure, refer to the OROUTED chapter in *z/OS Communications Server: IP Configuration Reference*. For more complete information about STDENV, refer to *z/OS Communications Server: IP User's Guide*.

Step 5: Specify the OROUTED Port Number in the SERVICES File

The services data set or file contains the relationship between service names (servers) and port numbers in the z/OS UNIX environment. The portion of the services file relevant to OROUTED is shown in Figure 66. The data set or file must exist for OROUTED to run. The following search order is used to find the services data set or file:

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

```
# Start of IBM added services ...
route        520/udp       router routed
```

Figure 66. Sample Portion of Services File

Step 6: Configure the Gateways File or Data Set (Optional)

The OROUTED server queries the network and dynamically builds routing tables from routing information transmitted by other routers that are directly connected to the network. The gateways file or data set is used to further configure the routing tables.

Note: The gateways file or data set is not related to the GATEWAY statement used in the PROFILE.TCPIP data set.

The OROUTED server uses the following search order to locate the GATEWAYS configuration data set or file:

1. If the environment variable GATEWAYS_FILE has been defined, OROUTED uses this value as the name of an MVS data set (//'mvs.dataset.name') or HFS file (/dir/subdir/file.name) to access the gateways file
2. /etc/gateways
3. *hlq*.ETC.GATEWAYS

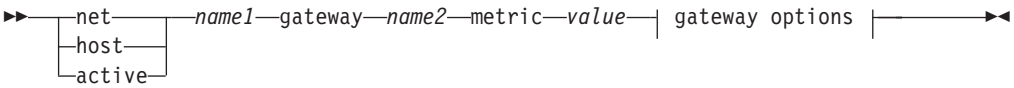
A sample gateways file is provided in *hlq*.SEZAINST(EZARTGW).

A `passive` entry in the gateways file or data set is used to add a route to a part of the network that does not support RIP. An `external` entry in the gateways file or data set indicates a route that should never be added to the routing tables. If another RIP server offers this route to your host, the route is discarded and not added to the routing tables. An `active` entry indicates a gateway that can only be reached through a network that does not allow or support link-level broadcasting or multicasting.

Syntax Rules

- The maximum LRECL allowed for the ETC.GATEWAY dataset is 999.
- Keywords can be specified in mixed case.
- Blanks and comments are supported in the gateways file or data set. Comments are identified by a semicolon in column 1.
- There should be no sequence numbers in the data set.

The syntax for the gateways file or data set is:



gateway options:

net
Indicates the route goes to a network.

host
Indicates the route goes to a specific host.

active
Indicates that the route to the gateway will be treated as a network interface. Active gateways are routers that are running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

name1
Can be either a symbolic name or the IP address of the destination network or host. If an IP address is specified, it must be in the standard dotted decimal notation. All numbers will be interpreted as decimal values only. No hexadecimal or octal notation will be accepted.

name1 must be specified as “active” if this is for an active gateway. The last entry in the data set must specify an active gateway.

gateway
A constant. The parameters that follow this keyword identify the gateway or router for this destination.

name2
Can be either a symbolic name or the IP address of the gateway or router for this destination. If an IP address is specified, it must be in the standard dotted decimal notation. All numbers will be interpreted as decimal values only. No hexadecimal nor octal notation will be accepted.

metric
A constant. The value that follows this keyword is the hop count to the destination host or network.

value
The hop count to this destination. This number is an integer in the range of 0 through 16, where 16 (infinity) indicates the network cannot be reached.

passive
A passive gateway does not exchange routing information. Information about the passive gateway is maintained in the local routing tables indefinitely and is only local to this OROUTED server. Passive gateway entries for indirect routes are not included in any routing information that is transmitted. Directly connected passive routes are included.

external
An external gateway parameter indicates that entries for this destination should never be added to the routing table. The OROUTED server discards any routes for this destination that it receives from other routers. Only the destination field is significant. The gateway parameter is ignored, but you must specify a routing interface in the network. The metric field is ignored.

active
Active gateways are treated as network interfaces. Active gateways are routers that are running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

mask
A constant. The value that follows this keyword is the subnet mask for the route.

subnetmask
A bit mask (expressed in dotted-decimal form) defining the subnetwork mask for a network route. The bits must be contiguous in the network portion of the *subnetmask*. If the *subnetmask* is not specified, OROUTED will default the subnetwork mask to an interface subnetwork mask that matches the route's network. If there is no interface match, then the network class mask for the route is used.

Note: For more information on passive, external, and active gateways, see *z/OS Communications Server: IP Configuration Reference*.

The following example shows the contents of a gateways file or data set containing multiple entries:

```
net    acmenet       gateway  gateway.acme.com  metric 5  passive
host   vm3.ibm.com   gateway  9.67.43.126       metric 6  passive
host   bad.host      gateway  xxx               metric 1  external
active active        gateway  9.3.1.110         metric 3  active
net    0.0.0.0       gateway  9.67.112.1        metric 1  passive
```

In the first entry, the route indicates that `acmenet` can be reached through the gateway `gateway.acme.com`, and that it is 5 hops away.

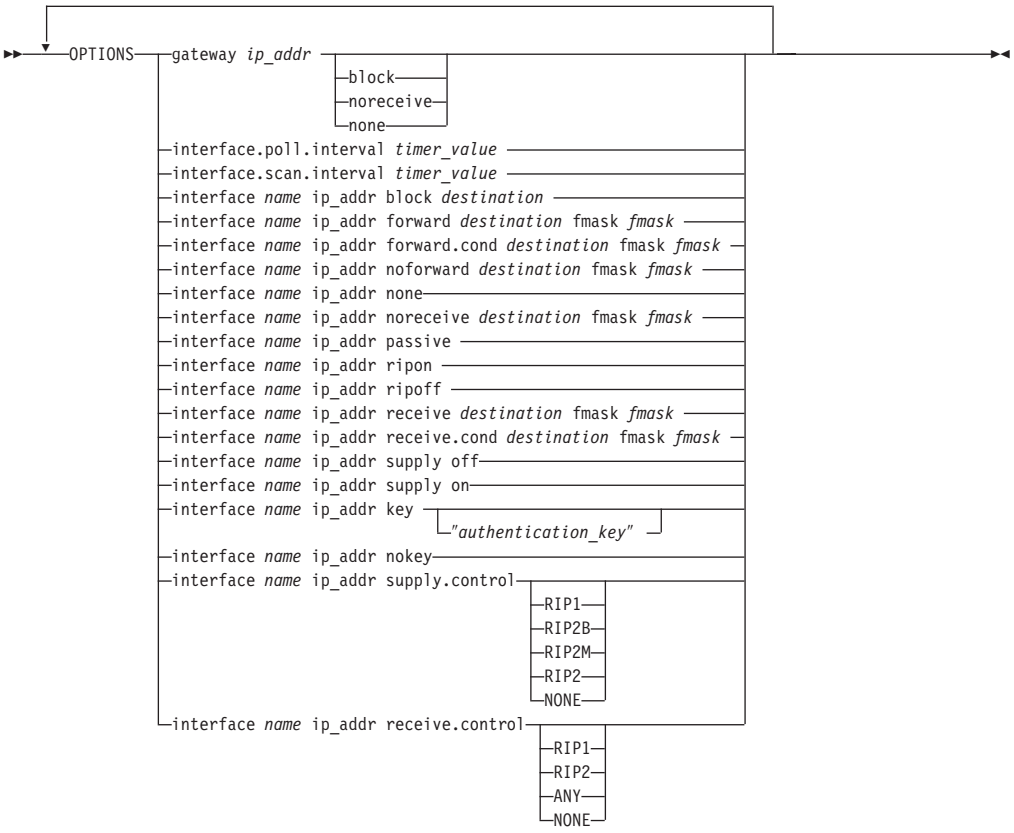
In the second entry, the route indicates that `vm3.ibm.com` can be reached through the gateway `9.67.43.126`, and that it is 6 hops away.

In the third entry, the external gateway parameter indicates that routes for the host `bad.host` should not be added to the routing tables, and that routes received from other OROUTED servers for `bad.host` should not be accepted.

The fourth entry shows an active gateway.

The fifth entry shows a default route to the destination gateway 9.67.112.1.

The syntax for the OPTIONS statement for the gateways file or data set is:



gateway
A constant. The value that follows this keyword identifies the gateway or router.

interface.scan.interval
Specifies the time interval in seconds for the interface scan interval. OROUTED uses this timer value to rescan existing interfaces for up/down status, new interfaces, and new HOME lists. New interfaces and HOME lists are dynamically added using VARY TCPIP,,CMD=OBEYFILE commands.
timer_value
The range is from 30 to 180 seconds in multiples of 30 seconds. The default is 60 seconds.

interface.poll.interval
Specifies the time interval in seconds for the interface poll interval. OROUTED uses this timer value to check existing interfaces for up/down status only.

Triggered updates are issued during interface outages to inform adjacent routers of unreachable routes so that alternative routes can be discovered.

timer_value
The range is from 15 to 180 seconds in multiples of 15 seconds. The default is 30 seconds.

interface
A constant

name
Specifies the name of the interface as used in the HOME list. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all interface names.

ip_addr
Specifies the internet address of the interface associated with the interface name. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all internet addresses of the interfaces.

block
For the interface option, specifies that the *destination* route in the received broadcasts for this interface is to be ignored. For the gateway option, specifies that routing table RIP responses from this gateway are to be ignored. This option is provided as a RIP input filter.

destination
Specifies the destination route in network, subnetwork, or host format. A specification of an asterisk (*) indicates that all destination routes to be used with the noforward and noreceive options. This serves as a ″blackhole″ filter option which can be used to filter out all routes from RIP packets to be sent or received over an interface and allow routes with specified forward and receive filters to be used.

fmask
Specifies the optional route filter mask.

forward
Specifies that the *destination* route in the RIP responses is to be forwarded to this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

forward.cond
Specifies that the *destination* route is to be forwarded to this interface only when the interface is active. In case of an interface outage, OROUTED will include the *destination* route in the RIP responses to other active interfaces. After recovery of an interface outage, ORouteD will resume to sending the destination route over this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

noforward
Specifies that the destination route in the RIP responses is not to be forwarded. This option is provided as a RIP output filter.

noreceive
See description for block.

passive
Same as ripoff.

receive
Specifies that the *destination* route is to be received over this interface only. If it is received over any other interface, the route is discarded. This option is provided as a RIP input filter.

receive.cond
Specifies that the *destination* route is to be received over this interface only when the interface is active. In case of an interface outage, OROUTED will allow the *destination* route in the RIP responses to be received over other active interfaces. This option is provided as a RIP input filter and can be used for inbound and outbound traffic splitting..

ripoff
Specifies that RIP is disabled for this interface. OROUTED will not send or receive RIP updates. This option is provided as a RIP input and output filter.

ripon
Specifies that RIP is enabled for the interface. This is the default for all interfaces. This option should be used when RIP has been previously disabled for an interface with the ripoff option, but is now required to be enabled for that interface.

supply off
Specifies that supplying RIP responses is disabled for this interface. OROUTED will not send, but continues to receive RIP responses. This option is provided as a RIP output filter.

supply on
Specifies that supplying RIP responses is enabled for this interface. This option is provided as a RIP output filter.

none
For the interface option, specifies that any RIP filter options for this interface are to be turned off or reset. If an asterisk (*) is specified for the interface *name* and *ip_addr*, all options will be cleared from all interfaces. For the gateway option, specifies that any RIP filter options for this gateway are to be turned off or reset. If an asterisk (*) is specified for the internet address, all gateway entries with gateway options will be cleared.

key
Specifies a plain text password containing up to 16 characters for the authentication key to be used for this interface and is used to override the router-wide setting defined in the OROUTED profile data set. The key must be enclosed in double quotes for the delimiters and can contain mixed case and blank characters. A no key or null key (″″) specification indicates that the router-wide key will be used as the default.

authentication_key
An authentication key containing up to 16 characters to be used for this interface and is used to override the OROUTED profile setting. The key must be enclosed in double qotes. The key will start with the first character past the first quotation mark and end at the last character before the last quotation mark on the line.

nokey
Specifies that authentication is disabled for this interface even though the router-wide specification from the OROUTED profile is defined.

supply.control
A constant. Specifies that the keyword follwing is to be used as the RIP supply control for this interface and is used to override the OROUTED profile setting. Possible supply controls are as follows:

RIP1 Unicast/Broadcast RIP Version 1 packets (Default)

RIP2B Unicast/Broadcast RIP Version 2 packets (Not Recommended)

RIP2M
Unicast/Multicast/Broadcast RIP packets (Migration)

RIP2 Unicast/Multicast RIP Version 2 packets

NONE Disable sending RIP packets

receive.control
A constant. Specifies that the variable follwing is to be used as the RIP receive control for this interface and is used to override the OROUTED profile setting. Possible receive controls are as follows:

RIP1 Receive RIP Version 1 packets

RIP2 Receive RIP Version 2 packets

ANY Receive any RIP Version 1 and 2 packets (Default)

NONE Disable receiving RIP packets

The following example shows the options entries of a gateways file or data set:

```
options interface.scan.interval 90
options interface.poll.interval 15
options interface ETH1 10.1.1.1 passive
options interface ETH1 10.1.1.1 supply off
options interface TR1 9.67.112.25 forward 11.0.0.0
options interface TR1 9.67.112.25 forward.cond 12.0.0.0
options interface TR1 9.67.112.25 block 9.1.0.0
options interface TR1 9.67.112.25 supply.control rip1
options interface ETH1 10.1.1.1 receive.control rip2
options interface ETH1 10.1.1.1 key
options interface CTC0  9.67.114.22 key "shredder"
options interface ETH1 10.1.1.1 none
options interface * * none
options gateway 9.2.1.4 noreceive
options gateway 9.2.1.4 none
options gateway * none
```

Step 7: Configure and Start syslogd

If not already started, configure and start syslogd. For more information on syslogd, see “Logging of System Messages” on page 25. Otherwise, all output messages will go to the operator system console.

Step 8: RACF-Authorize User IDs

To control which users can start OROUTED (and thus reduce risk of an unauthorized user starting it and affecting the contents of the routing table), the appropriate users must be RACF-authorized to the entity MVS.ROUTEMGR.OROUTED. To do this, the following commands must be entered from a RACF user ID, substituting the authorized user ID on the ID (userid) parameter:

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OROUTED) UACC(NONE)
PERMIT MVS.ROUTEMGR.OROUTED ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

OROUTED Parameters

OROUTED accepts the command line parameters listed below. These parameters are valid when starting the program from either an MVS procedure or from the z/OS shell. They are also valid when modifying OROUTED with the MODIFY command. For information on using the MODIFY command, see *z/OS Communications Server: IP Configuration Reference*.

-del
All dynamic routes are deleted from the routing table upon initialization of OROUTED. By default, OROUTED deletes only RIP routes at startup.

-d Enables printing internal debug information to STDOUT. This option should only be used to debug problems. When this option is specified, the -ep parameter is set internally.

-dp
Traces packets to and from adjacent routers and received and sent RIP packets. Packets are displayed in data format. Output is written to STDOUT.

-ep
Enable display of program print statements to STDOUT and STDERR. Information can be saved to a file by redirecting STDOUT to a file using the '>' operator. If this option is specified, and the program is started in the background from an z/OS shell, the userid will not be able to exit the shell until the program has ended.

-g Enables the default router. When this option is specified, OROUTED will add a default route to its routing information and propagate it over all local interfaces. If the adjacent routers add the default route to their routing tables, OROUTED will receive all unknown packets from them and funnel them to a destination router, provided that a default route is defined. If you use this option, we recommend that you define a default route to a destination router in the gateways file or data set. See "Configuring a Default Route" on page 579.

Note: Do not use this option if default routes are to be learned dynamically from adjacent routers.

-h Include host routes in addition to network routes for the RIP responses. Adjacent routers must be able to receive host routes to prevent NETWORK UNREACHABLE problems from occurring.

-hv
Include only VIPA host routes in addition to network routes for the RIP responses. Adjacent routers must be able to receive host routes' otherwise, network or subnetwork portions of VIPA addresses must be unique for each z/OS TCP/IP stack.

-q Suppresses supplying routing information.

-sd
Supply default route only. When this option is specified, the -g parameter is set internally. This option is provided as a RIP output filter.

-sdv (or -svd)
Supply network-specific VIPA routes and default routes only. See parameter descriptions for -sv and -sd. This option is provided as a RIP output filter.

-sl Supply local (directly-connected) routes only. This option is provided as a RIP output filter.

-st Supply triggered updates only. Similar to the -q parameter except that OROUTED will supply network unreachable routing information during interface outages so that adjacent routers can recover by switching to different routes rather than relying on three-minute timeouts. This option is provided as a RIP output filter.

-sv
Supply network-specific VIPA routes only. Recommended usage is when multiple network adapters in a z/OS TCP/IP stack are in the same network; otherwise, network connectivity problems will occur. This option is provided as a RIP output filter.

-svd
Similar to -sdv parameter.

-svh (or -shv)
Supply VIPA (network-specific and host) routes only. This option is provided as a RIP output filter.

-t Activates tracing of actions by the OROUTED server.

-t -t
Activates tracing of actions and packets sent or received.

-t -t -t
Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced and are displayed whenever an interface becomes inactive.

-t -t -t -t
Activates tracing of actions, packets sent or received, packet history, and packet contents. The packet displays the RIP network routing information.

Table 24 shows how the above parameters affect the advertising algorithm for routes in RIP responses to adjacent routers. The parameters can be used as router-wide RIP output filters. To configure interface-wide RIP input and output filters, see the OPTIONS statement in the GATEWAYS configuration data set or file.

Table 24. ORouteD Parameters

Parameter	Host Routes	VIPA Host Routes	Network Routes	VIPA Network Routes	Advertise as Default Router	Local Routes	Unreachable Routes
-g			Yes	Yes	Yes	Yes	Yes
-h	Yes	Yes	Yes	Yes		Yes	Yes
-hv		Yes	Yes	Yes			Yes
-s			Yes	Yes		Yes	Yes
-sd					Yes		Yes
-sl						Yes	Yes
-sq or -q							
-st							Yes
-sv				Yes			Yes
-svd				Yes	Yes		Yes

Table 24. ORouteD Parameters (continued)

-svh		Yes		Yes			Yes
None			Yes	Yes		Yes	Yes

Specifying Parameters

If OROUTED is to be started from an MVS procedure, add your parameters to PARM= in the OROUTED cataloged procedure. For example:

```
//OROUTED  PROC //OROUTED  EXEC PGM=OROUTED,REGION=4096K,TIME=NOLIMIT, // PARM=('POSIX(ON)', //
```

If OROUTED is to be started from an z/OS shell command line, enter the parameters on the z/OS shell command line.

For either method of starting OROUTED, the following apply:

- Each parameter is separated by a blank.
- Parameters can be specified in mixed case.

Starting OROUTED

In a CINET environment, OROUTED will attempt to connect to a stack name whose name is determined by the *TCPIPJobname* keyword from the resolver configuration data set or file. The *TCPIPJobname* must match the NAME field for ENTRYPOINT(EZBPFINI) in the BPXPRMxx member you used to start OMVS. For information on configuring mulitple TCP/IP instances as z/OS UNIX CINET physical file systems, see “Considerations for Multiple Instances of TCP/IP” on page 37.

In configurations with multiple stacks, a copy of OROUTED must be started for each stack that requires OROUTED services. To associate OROUTED with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique *TCPIPjobname*. A unique gateways file can be associated with each copy of OROUTED by defining the environment variable GATEWAYS_FILE. In the example in Figure 67 for the z/OS shell, there are two active stacks with the names TCP_A and TCP_B. First the environment variable RESOLVER_CONFIG is set to point to the file that defines the *TCPIPjobname* TCP_A, then the environment variable GATEWAYS_FILE is set to point to the gateways file /etc/gateways.tcp_a, and then OROUTED is started for that stack. Next, RESOLVER_CONFIG is set to point to another configuration file that defines *TCPIPjobname* TCP_B, then the environment variable GATEWAYS_FILE is set to point to the gateways file /etc/gateways.tcp_b, and then OROUTED is started for that stack.

```
#  export RESOLVER_CONFIG=/u/user105/tcp_a.conf            !point to TCP_A resolver file
#  export GATEWAYS_FILE=/etc/gateways.tcp_a                !point to TCP_A gateways file
#  export _BPX_JOBNAME=RDA                                 !set procname to RDA
#  export ROUTED_PROFILE=/u/user105/rda.profile            !set routed profile
#  orouted&                                                !start orouted in the background
#  export RESOLVER_CONFIG=/u/user105/tcp_b.conf            !point to TCP_B resolver file
#  export GATEWAYS_FILE=/etc/gateways.tcp_b                !point to TCP_77 bB gateways file
#  export _BPX_JOBNAME=RDB                                 !set procname to RDB
#  export ROUTED_PROFILE=/u/user105/rda.profile            !set routed profile
#  orouted&                                                !start orouted in the background
```

Figure 67. Example Commands to Start Multiple Copies of OROUTED

When running from an MVS procedure, the environment variables can be set by using the STDENV DD statement in the procedure used to start OROUTED. For an example of using the STDENV DD statement, see “OROUTED Cataloged Procedure” on page 567.

Configuration Examples

This section contains examples for configuring the OROUTED server. The following example illustrates a OROUTED configuration.

Configuring a Passive Route

In Figure 68, assume that your z/OS server is `host1` and is running an OROUTED server. The other two hosts, `host2` and `host3`, are not running a RIP server. Your OROUTED server does not learn a route to `host3`, because `host2` is not running a RIP server. Your OROUTED server sends routing updates to `host3` over the link to `host2` but never receives a routing update from `host2`. After 180 seconds, your OROUTED server deletes the route to `host2`. This problem is inherent to the RIP protocol and cannot be prevented.

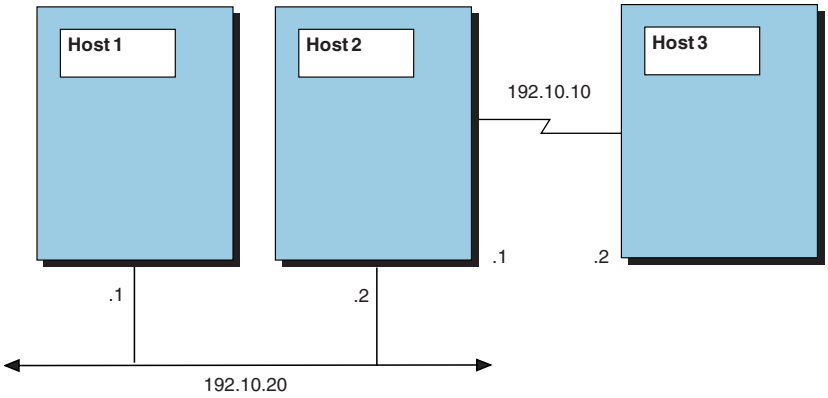


Figure 68. OROUTED Configuration Example

To solve the problem, you should add a `passive` route to this host in the gateways file or data set. You can use either of the following gateway statements:

```
host  host3        gateway  host2        metric 2  passive

host  192.10.10.2  gateway  192.10.20.2  metric 2  passive
```

Similarly, if `host2` is not running a RIP server, you can define a directly-connected `passive` route as follows:

```
host  host2        gateway  host1        metric 1  passive
```

A directly-connected passive route is one where the gateway address or name is one of the local interfaces in the HOME list.

Assume that your z/OS server is now `host2` and is running a OROUTED server. `host1` is also running a RIP server, but `host3` is not. Your OROUTED server sends routing information updates to `host3` over the link to `host3` but never receives a routing update from `host3`. After 180 seconds, your OROUTED server deletes the route to `host3`.

You should add a passive route to this host as follows:

```
host  host3        gateway  host2        metric 1  passive
```

host1 cannot reach host3 unless a passive routing entry is added to host1. For example:

```
host   host3    gateway  host2  metric  2  passive
```

or

```
host  192.10.10.2  gateway  192.10.20.2  metric  2  passive
```

Configuring an External Route

In Figure 68, assume that your z/OS server is again host1, which is running an OROUTED server. The other two hosts, host2 and host3, are also running RIP servers. Your OROUTED server normally learns a route to host3 from host2, because host2 is running a RIP server. You might not want host1 to route to host3 for security reasons. For example, a university might want to prevent student hosts from routing to administrative hosts.

To prevent your OROUTED server from adding a route to host3, add an external route to the gateways file or data set. You can use either of the following gateway statements:

```
host  host3        gateway  host2        metric 2  external

host  192.10.10.2  gateway  192.10.20.2  metric 2  external
```

Configuring an Active Gateway

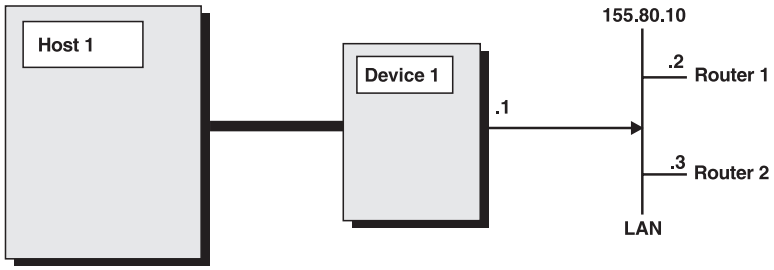


Figure 69. Configuring an Active Gateway

As shown in Figure 69, assume that your MVS host is host1, which is running an OROUTED server and that device1 is a network attachment device that does not support link-level broadcasting or multicasting or one that does not support ARP processing (for example, HYPERchannel). Also, assume that host1 is channel-connected to device1 as a hyperchannel device and that there are routers router1 and router2 on the local area network. Because the IP addresses for router1 and router2 are unknown by host1, they have to be manually configured in host1 for OROUTED to communicate with them. Configuring active gateways for router1 and router2 as remote network interfaces enables OROUTED to send RIP responses to the target addresses.

Include the following definitions in *hlq*.PROFILE.TCPIP for host1, router1, and router2:

1. Specify DEVICE and LINK statements for the hyperchannel. For example:

```
DEVICE HYPER1   HCH    CE2
LINK   HYPER1A  HCH    2     HYPER1A
```

2. Add the TRANSLATE statement for the remote routers on the local area network attached to the hyperchannel device:

```
TRANSLATE 155.80.10.2 HCH    HYPER1A
TRANSLATE 155.80.10.3 HCH    HYPER1A
```

3. Add the hyperchannel link to the HOME statement for the assignment of local IP address:

```
HOME
     155.80.10.1  HYPER1A
```

4. Add the hyperchannel link to the BSDROUTINGPARMS statement:

```
BSDROUTINGPARMS false
      HYPER1A       16384      0      255.255.240.0    0
ENDBSDROUTINGPARMS
```

Define active gateways for the remote routers in the OROUTED gateways file or data set:

```
active active gateway 155.80.10.2 metric 1 active
active active gateway 155.80.10.3 metric 1 active
```

From these active gateway addresses, OROUTED will use them as the destination addresses to send RIP responses to the remote routers. In addition, OROUTED will continue to receive RIP responses from the active gateways over the hyperchannel device.

Configuring a Point-to-Point Link

The OROUTED server can manage point-to-point links as long as there are RIP services at both ends of the links. If a host router at the other end of a link is not running a RIP service, then passive routing must be configured in the OROUTED gateways data set for the link. See “Configuring a Passive Route” on page 577.

Configuring a Default Route

A default route is typically used on a gateway or router to an internet, or on a gateway or router that uses another routing protocol, whose routes are not reported to other local gateways or routers.

To configure a route to a default destination, add a default route using the passive route definition in the gateways file or data set. For example, if the default destination router has a gateway address 9.67.112.1, then add the following entry to the data set:

```
net  0.0.0.0  gateway  9.67.112.1  metric  1  passive
```

Only one default route to a destination gateway or router can be specified. OROUTED currently does not support multiple default routes.

Configuring ORouteD with Enterprise Extender

If running ORouteD with the Enterprise Extender on a z/OS TCP/IP stack connected to SNA via IUTSAMEH device, special configuration needs to be considered depending on the system environment. Assume that TOVTAM interface is the link name for the IUTSAMEH device and its home IP address is 9.2.1.1.

1. If there are no other z/OS TCP/IP stacks in this MVS image, disable RIP on the TOVTAM interface to prevent ORouteD from sending any routing information over this interface. For example, in the ORouteD gateways file or data set:

   ```
   options interface TOVTAM 9.2.1.1 ripoff
   ```

2. If one or more z/OS TCP/IP stacks exist in the MVS image, do the following:

- Define the link characteristics for the TOVTAM interface in the BSDROUTINGPARMS statement of TCP/IP configuration profile. In this definition, specify a subnet mask and a zero IP destination address.

  For example, assume that the subnet mask for the TOVTAM interface is 255.255.255.0:

  ```
  ; link  maxmtu       metric   subnet mask     dest_addr
  TOVTAM  DEFAULTSIZE    0     255.255.255.0       0
  ```

  If using RIP1 or RIP2B on this interface and depending on its subnet mask, ORouteD will use a subnet-directed broadcast address to send the routing information to other z/OS TCP/IP stacks in this MVS image. If using RIP2 or RIP2M, ORouteD will use the RIP2 multicast address.

3. Repeat the above step for other z/OS TCP/IP stacks running with ORouteD and configured with IUTSAMEH devices in this MVS image. Ensure that the RIP settings for the TOVTAM interfaces are identical so that the routing information exchanges will occur between z/OS TCP/IP stacks running ORouteD in this MVS image.

Configuring OROUTED with VIPA

For more information on configuration options with VIPA, see the following topics:

- “Chapter 3. Virtual IP Addressing” on page 109
- “Configuring Static VIPAs for an z/OS TCP/IP Stack” on page 113
- “Planning for Static VIPA Takeover and Takeback” on page 114
- “Configuring OROUTED to Split Traffic with VIPA”

Configuring OROUTED to Split Traffic with VIPA

The purpose of splitting traffic is to reduce traffic load on network attachments by controlling the inbound and outbound traffic. The following techniques can be used to produce traffic splitting effects with fault tolerance benefit:

- Using Interface Metric and VIPA To Split Inbound/Outbound Traffic

  In the multiple network attachments to the same network configuration, split inbound/outbound traffic can be achieved by configuring the metric on the primary interface to one higher than the secondary interfaces. From routing updates, an adjacent router uses the gateway of a secondary interface to reach the destination VIPA on the z/OS server because the route to the gateway has a shorter metric. The primary interface is used for outbound traffic and a secondary interface is used for inbound traffic. The traffic splitting will function as long as the primary and at least one secondary interfaces are active. For information on configuring an interface metric, see the *z/OS Communications Server: IP Configuration Reference*. A VARY TCPIP,,CMD=OBEYFILE command for the BSDROUTINGPARMS statement can be used to update an interface metric for a link. For an example of configuring a virtual device, see “Configuring OROUTED with VIPA”.

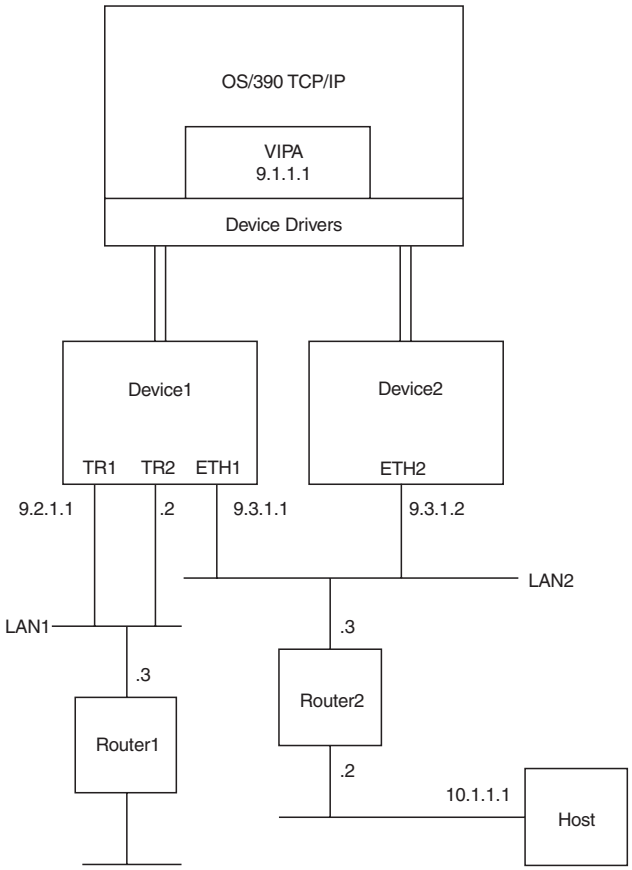


Figure 70. Single VIPA Configuration. Sample configuration showing primary/multiple network attachments to the same LAN, VIPAs, and inbound/outbound traffic splitting.

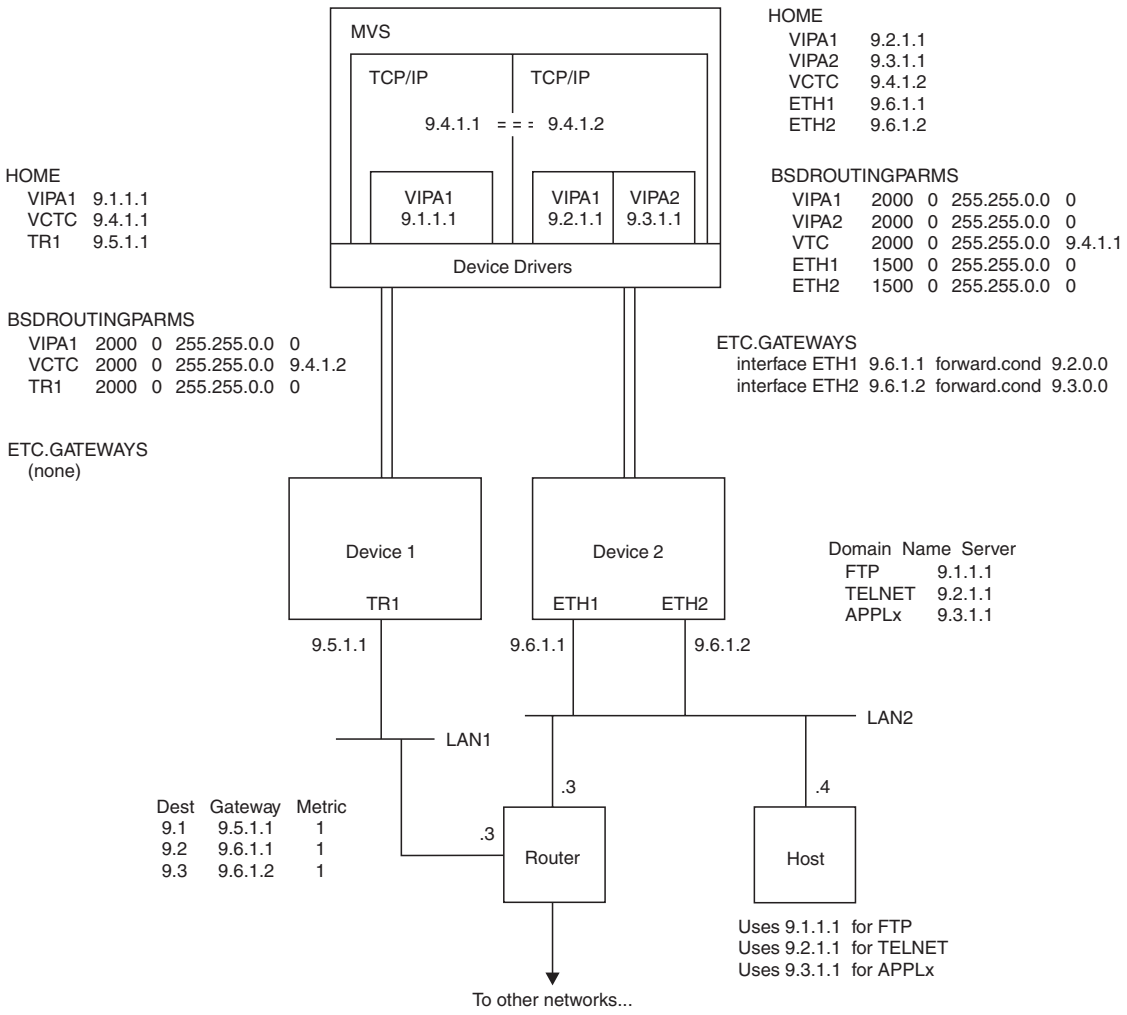


Figure 71. Multiple VIPA Configuration. Sample configuration showing primary/multiple network attachments to the same LAN, VIPAs, and inbound/outbound traffic splitting.

- Using Route Forwarding and VIPA to Split Session Traffic

  With multiple VIPAs in one TCP/IP stack, a VIPA can be assigned to a particular interface so that the VIPA can be reserved for session traffic (for example, FTP or TELNET). This is accomplished by using the route forwarding option in OROUTED. From routing updates, an adjacent router will have multiple gateways to reach the VIPAs on the z/OS 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/OS server. For fault tolerance, it is recommended that the conditional option of route forwarding be used. For information on route forwarding, see the options statement in “Step 6: Configure the Gateways File or Data Set (Optional)” on page 567. For an example of configuring a virtual device, see “Configuring OROUTED with VIPA” on page 580.

Appendix D. Information Apars

This appendix lists information apars for IP-related books.

Notes:

1. Information apars contain updates to previous editions of the manuals listed below. Books updated for V2R10 contain all the updates except those contained in the information apars that may be issued after V2R10 books went to press.
2. CS for OS/390 V2R10 information apars will contain updates for z/OS CS V1R1.
3. Information apars are predefined for CS for OS/390 V2R10 and may not contain updates.

IP Information Apars

Table 25 lists information apars for IP-related books.

Table 25. IP Information Apars

Title	CS for OS/390 2.10	CS for OS/390 2.8	CS for OS/390 2.7	CS for OS/390 2.6	CS for OS/390 2.5	TCP/IP 3.3
High Speed Access Service User's Guide (GC31-8676)		ii11629	ii11566	ii11412	ii11181	
IP API Guide (SC31-8516)	II12371	ii11635	ii11558	ii11405	ii11144	
IP CICS Sockets Guide (SC31-8518)		ii11626	ii11559	ii11406	ii11145	
IP Configuration (SC31-8513)		ii11620 ii12068	ii11555 ii11637 ii11995	ii11402 ii11619 ii12066	ii11159 ii11979	ii10633
IP Configuration Guide (SC31-8725)	II12362					
IP Configuration Reference (SC31-8726)	II12363					
IP Diagnosis (SC31-8521)	II12366	ii11628	ii11565	ii11411	ii11160 ii11414	ii10637
IP Messages Volume 1 (SC31-8517)	II12367	ii11630	ii11562	ii11408		Messages and Codes ii10635
IP Messages Volume 2 (SC31-8570)	II12368	ii11631	ii11563	ii11409		
IP Messages Volume 3 (SC31-8674)	II12369	ii11632	ii11564	ii11410	ii11158	

Table 25. IP Information Apars (continued)

Title	CS for OS/390 2.10	CS for OS/390 2.8	CS for OS/390 2.7	CS for OS/390 2.6	CS for OS/390 2.5	TCP/IP 3.3
IP Migration (SC31-8512)	II12361	ii11618	ii11554	ii11401		
IP Network Print Facility (SC31-8522)		ii11627	ii11561	ii11407	ii11150	
IP Programmer's Reference (SC31-8515)		ii11634	ii11557	ii11404		ii10636
IP and SNA Codes (SC31-8571)	II12370	ii11917	Added TCP/IP codes to VTAM codes V2R6 ii11611	ii11361	ii11146 ii11097	
IP User's Guide (GC31-8514)	II12365	ii11625	ii11556	ii11403	ii11143	ii10634

Appendix E. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O.Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

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

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

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

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

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

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs

conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Release 1 Collection (SK3T-4269-00), which contains the unlicensed books for z/OS in BookManager and PDF format, or the z/OS Internet Library (http://www.ibm.com/s390/os390/). To order the latest hardcopy edition that is available, you might need to order a lower suffix (dash) level.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Peer-to-Peer Networking
AFP
AD/Cycle
AIX
AIX/ESA
AnyNet
APL2
APPN
AS/400
AT
BookManager
BookMaster
CBPDO
C/370
CICS
CICS/ESA
C/MVS
Common User Access
C Set ++
CT
CUA
DATABASE 2
DatagLANce
DB2
DFSMS
DFSMSdfp
DFSMShsm
DFSMS/MVS
Domino
DRDA
eNetwork
Enterprise Systems Architecture/370
ESA/390
ESCON
ES/3090
ES/9000
ES/9370
EtherStreamer
Extended Services
FAA
MVS
MVS/DFP
MVS/ESA
MVS/SP
MVS/XA
MQ
Natural
NetView
Network Station
Nways
Notes
NTune
NTuneNCP
OfficeVision/MVS
OfficeVision/VM
Open Class
OpenEdition
OS/2
OS/390
Parallel Sysplex
Personal System/2
PR/SM
PROFS
PS/2
RACF
Resource Measurement Facility
RETAIN
RFM
RISC System/6000
RMF
RS/6000
S/370
S/390
SAA
SecureWay
Slate
SP
SP2
SQL/DS
System/360
System/370

FFST
FFST/2
FFST/MVS
First Failure Support Technology
GDDM
Hardware Configuration Definition
IBM
IBMLink
IMS
IMS/ESA
InfoPrint
Language Environment
LANStreamer
Library Reader
LPDA
MCS
Micro Channel
System/390
SystemView
Tivoli
TURBOWAYS
UNIX System Services
Virtual Machine/Extended Architecture
VM/ESA
VM/XA
VSE/ESA
VTAM
WebSphere
XT
z/OS
400
3090
3890

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

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

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see http://www.intel.com/tradmarx.htm.

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

Bibliography

z/OS Communications Server Publications

This bibliography contains descriptions of the books in the z/OS Communications Server library.

z/OS Communications Server publications are available online at http://www.ibm.com/servers/eserver/zseries/zos/ the z/OS Internet Library web page.

Our information is available in hard- and soft-copy, soft-copy only, or web-only. The following abbreviations follow each order number:

HC/SC — both hard- and soft-copy are available.

SC — only soft-copy is available. These books are available on the CD Rom accompanying z/OS (SK3T-4269) or at the z/OS internet library site.

WEB — these books are available at the following web site: http://www.ibm.com/s390/os390/bkserv/ or on SK2T-6700.

Updates to books are available on RETAIN and in the document called *OS/390 DOC APARs and ++HOLD DOC data* which can be found at http://www.s390.ibm.com/os390/bkserv/new_tech_info.html. See “Appendix D. Information Apars” on page 583 for a list of the books and the INFOAPARS associated with them.

Softcopy Information

- *z/OS V1R1 Collection* (SK3T-4269–00).

  This is the CD collection shipped with the z/OS product. It includes the libraries for z/OS V1R1, in both BookManager and PDF formats.
- *z/OS Software Products Collection* (SK3T-4270).

  This CD includes the libraries of z/OS software products that run on z/OS but are not elements and features, in both BookManager and PDF formats, as well as the Getting Started with Parallel Sysplex bookshelf.
- *z/OS V1R1 and Software Products DVD Collection* (SK3T-4271).

  This collection includes the libraries of z/OS (the element and feature libraries) and the libraries for z/OS software products in both BookManager and PDF format. This collection combines SK3T-4269 and SK3T-4270.
- *z/OS Licensed Product Library* (SK3T-4307).

  This CD includes the licensed books in both BookManager and PDF format.
- *OS/390 Online Library Collection* (SK2T-6700).

  This collection contains softcopy unlicensed books for OS/390, Parallel Sysplex products, and S/390 application programs that run on OS/390.
- *OS/390 PDF Library Collection* (SK2T-6718).

  This collection contains the unlicensed books for OS/390 Version 2 Release 6 in Portable Document Format (PDF).
- *OS/390 Licensed Product Library* (LK2T-2499).

  This library contains unencrypted softcopy licensed books for OS/390 Version 2. The OS/390 Licensed Product Library for Version 1 (LK2T-6702) is still available, but is no longer updated.
- *System Center Publication IBM S/390 Redbooks Collection* (SK2T-2177).

  This collection contains over 300 ITSO redbooks that apply to the S/390 platform and to host networking arranged into subject bookshelves.

Planning

z/OS Communications Server: SNA Migration (GC31-8774 [HC/SC]). This book is intended to help you plan for SNA, whether you are migrating from a previous version or installing SNA for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with SNA.

z/OS Communications Server: IP Migration (GC31-8773 [HC/SC]). This book is intended to help you plan for IP, whether you are migrating from a previous version or installing IP for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with IP.

Resource Definition, Configuration, and Tuning

z/OS Communications Server: IP Configuration Guide (SC31-8775 [HC/SC]). This book describes the major concepts involved in understanding and

configuring an IP network. Familiarity with MVS operating, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this book in conjunction with the *z/OS Communications Server: IP Configuration Reference*.

z/OS Communications Server: IP Configuration Reference (SC31-8776 [HC/SC]). This book presents information for people who want to administer and maintain IP. Use this book in conjunction with the *z/OS Communications Server: IP Configuration Guide*. The information in this book includes:

- TCP/IP configuration data sets
- Configuration statements
- Operator commands
- Translation tables
- SMF records
- Protocol number and port assignments

z/OS Communications Server: SNA Network Implementation Guide (SC31-8777 [HC/SC]). This book presents the major concepts involved in implementing a SNA network. Use this book in conjunction with the *z/OS Communications Server: SNA Resource Definition Reference*.

z/OS Communications Server: SNA Resource Definition Reference (SC31-8778 [HC/SC]). This book describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA.Use this book in conjunction with the *z/OS Communications Server: SNA Network Implementation Guide*.

OS/390 eNetwork Communications Server: SNA Resource Definition Samples (SC31-8566 [WEB]). This book contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.

OS/390 eNetwork Communications Server: AnyNet SNA over TCP/IP (SC31-8578 [WEB]). This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.

OS/390 eNetwork Communications Server: AnyNet Sockets over SNA (SC31-8577 [WEB]). This guide provides information to help you install, configure, use, and diagnose Sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.

Operation

z/OS Communications Server: IP User's Guide (SC31-8780 [HC/SC]). This book is for people who want to use TCP/IP for data communication activities such as FTP and Telnet. Familiarity with MVS operating system and IBM Time Sharing Option (TSO) is recommended.

z/OS Communications Server: SNA Operation (SC31-8779 [HC/SC]). This book serves as a reference for programmers and operators requiring detailed information about specific operator commands.

z/OS Communications Server: Quick Reference (SX75-0124 [HC/SC]). This book contains essential information about SNA and IP commands.

Customization

z/OS Communications Server: SNA Customization (LY43-0092 [SC]). This book enables you to customize SNA, and includes:

- Communication network management (CNM) routing table
- Logon-interpret routine requirements
- Logon manager installation-wide exit routine for the CLU search exit
- TSO/SNA installation-wide exit routines
- SNA installation-wide exit routines

OS/390 eNetwork Communications Server: IP Network Print Facility (SC31-8522 [WEB]). This book is for system programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP.

Writing Application Programs

z/OS Communications Server: IP Application Programming Interface Guide (SC31-8788 [SC]). This book describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this book to adapt your existing applications to communicate with each other using sockets over TCP/IP.

OS/390 SecureWay Communications Server: IP CICS Sockets Guide (SC31-8518 [WEB]). This book is for people who want to set up, write

application programs for, and diagnose problems with the socket interface for CICS using z/OS TCP/IP.

OS/390 eNetwork Communications Server: IP IMS Sockets Guide (SC31-8519 [WEB]). This book is for programmers who want application programs that use the IMS TCP/IP application development services provided by IBM TCP/IP for MVS.

z/OS Communications Server: IP Programmer's Reference (SC31-8787 [SC]). This book describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the MVS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.

OS/390 IBM Communications Server: SNA Programming (SC31-8573 [WEB]). This book describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Guide (SC31-8581 [WEB]). This book describes how to use the SNA LU 6.2 application programming interface for host application programs. This book applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this book.)

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Reference (SC31-8568 [WEB]). This book provides reference material for the SNA LU 6.2 programming interface for host application programs.

OS/390 eNetwork Communications Server: CSM Guide (SC31-8575 [WEB]). This book describes how applications use the communications storage manager.

OS/390 IBM Communications Server: CMIP Services and Topology Agent Guide (SC31-8576 [WEB]). This book describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The book provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis

z/OS Communications Server: IP Diagnosis (GC31-8782 [HC/SC]). This book explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.

z/OS Communications Server: SNA Diagnosis V1 Techniques and Procedures(LY43-0088 [HC/SC]) and *z/OS Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* (LY43-0089 [HC/SC]). These books help you identify a SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.

z/OS Communications Server: SNA Data Areas Volume 1 (LY43-0090 [SC]) and *z/OS Communications Server: SNA Data Areas Volume 2* (LY43-0091 [SC]). These books describe SNA data areas and can be used to read a SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and Codes

z/OS Communications Server: SNA Messages (SC31-8790 [HC/SC]). This book describes the ELM, IKT, IST, ISU, IVT, IUT, and USS messages. Other information in this book includes:

- Command and RU types in SNA messages
- Node and ID types in SNA messages
- Supplemental message-related information

z/OS Communications Server: IP Messages Volume 1 (EZA) (SC31-8783 [HC/SC]). This volume contains TCP/IP messages beginning with EZA.

z/OS Communications Server: IP Messages Volume 2 (EZB) (SC31-8784 [HC/SC]). This volume contains TCP/IP messages beginning with EZB.

z/OS Communications Server: IP Messages Volume 3 (EZY) (SC31-8785 [HC/SC]). This volume contains TCP/IP messages beginning with EZY.

z/OS Communications Server: IP Messages Volume 4 (EZZ-SNM) (SC31-8786 [HC/SC]). This volume contains TCP/IP messages beginning with EZZ and SNM.

z/OS Communications Server: IP and SNA Codes (SC31-8791 [HC/SC]). This book describes codes and other information that display in z/OS Communications Server messages.

APPC Application Suite

OS/390 eNetwork Communications Server: APPC Application Suite User's Guide (GC31-8619 [WEB]). This book documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.

OS/390 eNetwork Communications Server: APPC Application Suite Administration (SC31-8620 [WEB]). This book contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.

OS/390 eNetwork Communications Server: APPC Application Suite Programming (SC31-8621 [WEB]). This book provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Redbooks

The following Redbooks may help you as you implement z/OS Communications Server.

- *IBM Communication Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 1: Configuration and Routing* (SG24–5227).
- *IBM Communication Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 2: UNIX Applications* (SG24–5228).
- *OS/390 eNetwork Communication Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications* (SG24–5229).
- *OS/390 Secureway Communication Server V2R8 TCP/IP Guide to Enhancements* (SG24-5631-00).
- *TCP/IP in a Sysplex* (SG24–5235–01).
- *SNA and TCP/IP Integration* (SG24–5291–00).
- *Managing OS/390 TCP/IP with SNMP* (SG24–5866–00).
- *Security in OS/390–based TCP/IP Networks* (SG24–5383–00).
- *TCP/IP in a Sysplex* (SG24–5335–01).
- *IP Network Design Guide* (SG24–2580–01).
- *SNA and TCP/IP Integration* (SG24–5291–00).

Related Publications

For information about z/OS products, refer to *z/OS Information Roadmap* (SA22-7500). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, as well as describing each z/OS publication.

Firewall

z/OS SecureWay Security Server Firewall Technologies (SC24-5922 [HC/SC])

OSA-Express

S/390: OSA-Express Customer's Guide and Reference (SA22-7403 [HC/SC])

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

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

z/OS Communications Server
IP Configuration Guide
Version 1 Release 1

Publication No. SC31-8775-00

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 _____

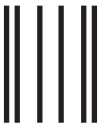
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC31-8775-00

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
Software Reengineering
Department G7IA/ Bldg 503
Research Triangle Park, NC
27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5694-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8775-00

Spine information:

z/OS Communications Server

z/OS V1R1.0 CS: IP Configuration Guide

Version 1
Release 1