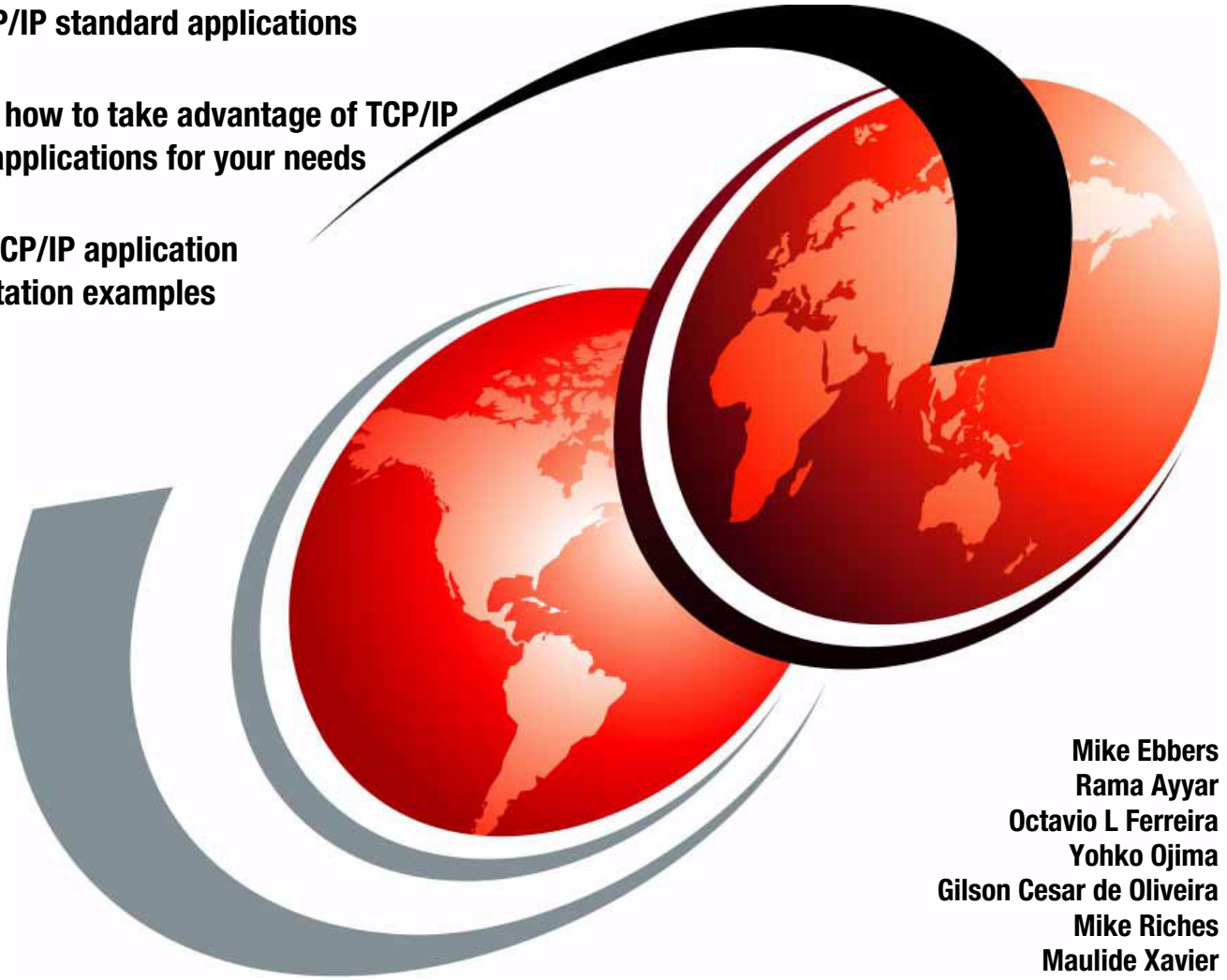
IBM

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 2 Standard Applications

- Provides information about z/OS Communications Server TCP/IP standard applications
- Discusses how to take advantage of TCP/IP standard applications for your needs
- Includes TCP/IP application implementation examples



Mike Ebbers
Rama Ayyar
Octavio L Ferreira
Yohko Ojima
Gilson Cesar de Oliveira
Mike Riches
Maulide Xavier

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 2 Standard Applications

December 2011

SG24-7997-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (December 2011)

This edition applies to Version 1, Release 13 of IBM z/OS Communications Server (product number 5694-A01).

Contents

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.

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.

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.

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 illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

AIX®
BladeCenter®
C/370™
CICS®
DB2®
Domino®
HiperSockets™
IBM®
IMS™
Language Environment®
Lotus Notes®
Lotus®
MVS™
NetView®
Notes®
Parallel Sysplex®
Print Services Facility™
PrintWay™
RACF®
Redbooks®
Redbooks (logo) ®
System z®
Tivoli®
VTAM®
WebSphere®
z/OS®
zEnterprise™

The following terms are trademarks of other companies:

PostScript, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

For more than 40 years, IBM® mainframes have supported an extraordinary portion of the world's computing work, providing centralized corporate databases and mission-critical enterprise-wide applications. The IBM System z®, the latest generation of the IBM distinguished family of mainframe systems, has come a long way from its IBM System/360 heritage. Likewise, its IBM z/OS® operating system is far superior to its predecessors, providing, among many other capabilities, world-class, state-of-the-art, support for the TCP/IP Internet protocol suite.

TCP/IP is a large and evolving collection of communication protocols managed by the Internet Engineering Task Force (IETF), an open, volunteer, organization. Because of its openness, the TCP/IP protocol suite has become the foundation for the set of technologies that form the basis of the Internet. The convergence of IBM mainframe capabilities with Internet technology, connectivity, and standards (particularly TCP/IP) is dramatically changing the face of information technology and driving requirements for ever more secure, scalable, and highly available mainframe TCP/IP implementations.

The *IBM z/OS Communications Server TCP/IP Implementation* series provides understandable, step-by-step guidance about how to enable the most commonly used and important functions of z/OS Communications Server TCP/IP.

This IBM Redbooks® publication provides useful implementation scenarios and configuration recommendations for many of the TCP/IP standard applications that z/OS Communications Server supports.

For more specific information about z/OS Communications Server standard applications, high availability, and security, see the other volumes in the series:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999

For comprehensive descriptions of the individual parameters for setting up and using the functions that we describe in this book, along with step-by-step checklists and supporting examples, see the following publications:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780

This book does not duplicate the information in those publications. Instead, it complements them with practical implementation scenarios that can be useful in your environment. To determine at what level a specific function was introduced, see *z/OS Communications Server: New Function Summary*, GC31-8771. For complete details, we encourage you to review the documents that are listed in the additional resources section at the end of each chapter.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Mike Ebbers is a Consulting IT Specialist and Project Leader at the International Technical Support Organization, Poughkeepsie Center. He has worked with IBM mainframe hardware and software products since 1974 in the field, in education, and in the ITSO.

Rama Ayyar is an independent IT Consultant based in Sydney, Australia and a former employee of IBM Australia. He has also worked for CSC Australia in senior technical roles. He is one of the founding members of HCL India. Rama has over 25 years of experience with the IBM MVS™ operating system. His areas of expertise include TCP/IP, RACF, DFSMS, z/OS, HCD/HCM, Dump Analysis, and Disaster Recovery. Rama has co-authored eight IBM Redbooks. He holds a Master's Degree in Computer Science from the Indian Institute of Technology, Kanpur. Rama has been in the computer industry for more than 35 years.

Octavio L. Ferreira is a Consulting IT Specialist with IBM in Brazil. He has 29 years of experience in IBM software support. His areas of expertise include z/OS Communications Server, SNA and TCP/IP, and Communications Server on all platforms. For the last 11 years, Octavio has worked at the Area Program Support Group, providing guidance and support to clients and designing networking solutions such as SNA-TCP/IP Integration, z/OS Connectivity, Enterprise Extender design and implementation, and SNA-to-APPN migration. He has also co-authored other IBM Redbooks publications.

Yohko Ojima is an Advisory IT Specialist in IBM Japan and has nine years of experience in enterprise networking. She specializes in TCP/IP and SNA networks with products including z/OS Communications Server, Communications Server for distributed servers, and Communication Controller for Linux on System z, and also devices such as routers and switches. Her responsibility is to provide technical consultations and services on networking design, implementation, and problem solving for customers.

Gilson Cesar de Oliveira is a Senior IT Support Specialist in Brazil working for HSBC as a System Programmer. He holds a degree in Computer Science and specialization in Data Networking. He has 20 years of experience in mainframe networking with expertise in VTAM/SUBAREA/APPN, NCP, TCP/IP, CS for z/OS, JES2/NJE, RACF/RRSF, IBM 3745/46, and OSA-Express.

Mike Riches is a Senior Network Specialist within Maintenance and Technical Support Services, United Kingdom, providing remote technical support and on-site consultancy for IBM clients across Europe. He has 20 years of experience working with IBM networking software, the last 9 of those working for IBM. His areas of expertise include z/OS Communications Server, SNA, APPN, HPR, EE and TCP/IP networking.

Maulide Xavier is an IT Specialist with IBM in Portugal. He has 13 years of experience in IBM mainframe and networking systems. His current responsibilities include network-related problem-solving on System z, and supporting clients to implement a network infrastructure in complex environments. His areas of expertise include z/OS Communications Server, SNA, and TCP/IP networking. He holds a diploma in Business and Economics Applied Math from Instituto Superior de Economia and Gestao and a post graduate degree in Management of Information Systems from the same school. He is also a member of IBM Networking Software Support in Europe.

Thanks to the following people for their contributions to this project:

David Bennin
Richard Conway
Robert Haimowitz
Bill White
International Technical Support Organization, Poughkeepsie Center

Andrew Arrowood
Michael Gerlach
Scott Moonen, the development sponsor
Alan Packett
IBM Communications Server Development, Raleigh

Thanks to the authors of the previous editions of this book:

Finally, we want to thank the authors of the previous *z/OS Communications Server TCP/IP Implementation* series for creating the groundwork for this series: Rama Ayyar, Valirio Braga, WenHong Chen, Demerson Cilloti, Gwen Dente, Gilson Cesar de Oliveira, Sandra Elisa Freitag, Mike Ebbers, Octavio L. Ferreira, Marco Giudici, Adi Horowitz, Michael Jensen, Gazi Karakus**,** Shizuka Katoh, Sherwin Lake, Bob Louden, Garth Madella, Yukihiko Miyamoto, Hajime Nagao, Shuo Ni, Carlos Bento Nonato, Yohko Ojima, Roland Peschke, Joel Porterie, Marc Price, Frederick James Rathweg, Micky Reichenberg, Larry Templeton, Rudi van Niekerk, Bill White, Thomas Wienert, Andi Wijaya, and Maulide Xavier.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

The syslog daemon

The syslog daemon (syslogd) is a server that writes messages from applications to log files. This chapter describes syslogd functions that are available in the z/OS Communications Server.

This chapter discusses the following syslogd topics.

Section	Topic
1.1, “Conceptual overview of syslogd” on page 2	The basic concepts of syslogd.
1.2, “Log messages to different files and to a single file” on page 5	How to configure syslogd to log messages to different files, based on job name, and also send all messages to a single file without regard to job name.
1.3, “Starting two syslogd instances” on page 14	How to configure and run two instances of syslogd in local mode and sin network mode.
1.4, “The syslogd functions” on page 18	How to ► Configure the sylogd.conf to perform automatic archival ► Use operator commands to control syslogd ► Configure the syslogd browser and search facility
1.5, “Problem determination for syslogd logging” on page 33	How to perform syslogd problem determination.
1.6, “Additional information sources for syslogd” on page 34	References to additional syslogd documentation materials.

1.1 Conceptual overview of syslogd

The syslogd is a standard application that ships with z/OS Communications Server. You use the syslogd to manage log records that are written by several applications, such as ftpd and omproute. The syslogd uses rules that are coded in a configuration file to determine the destination of each log record. Many destination types are supported; however, a common type of destination is a z/OS UNIX file.

Figure 1-1 illustrates the relationship of syslogd to applications and to other hosts for which it provides logging services. The syslogd running in local mode (**`syslogd -i`**) can collect messages from local applications and can either record them into a local file for that application or forward them to another remote host for logging.

The syslogd running in network mode (**`syslogd -n`**) accepts messages over the network from other systems and either records them into a local file or forwards them to yet another remote host for logging. If both types of syslogd are running, each can have its own configuration file, or they can share a common configuration.

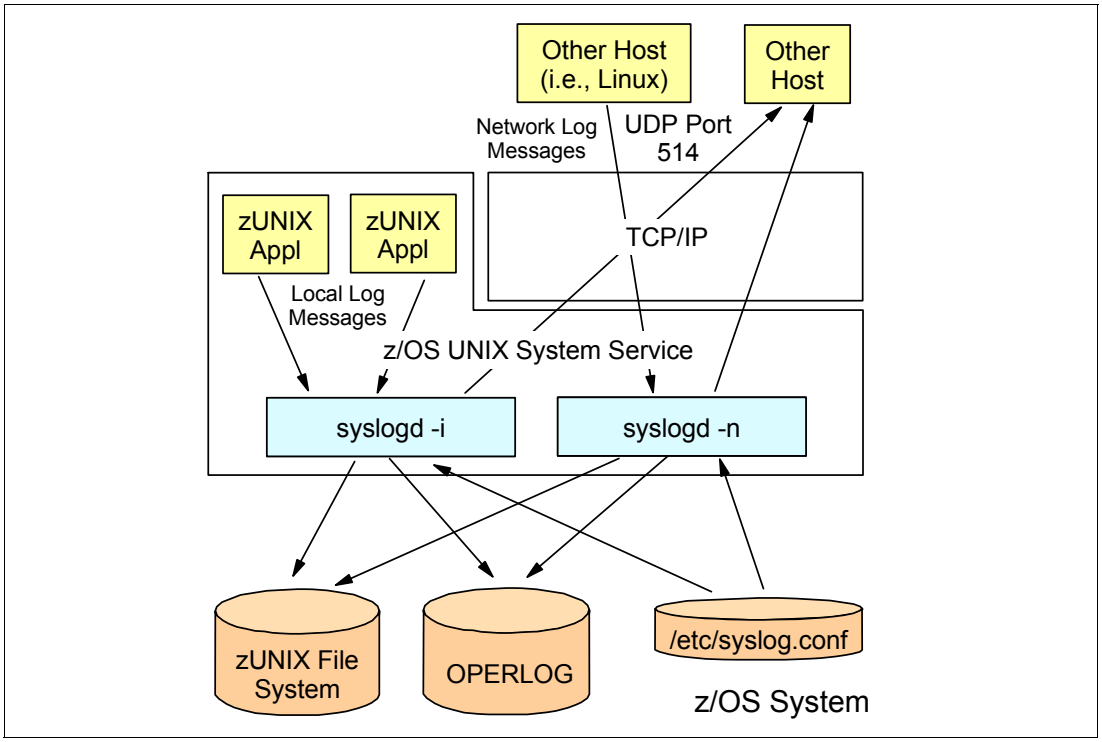


Figure 1-1 The syslogd relationship to applications for which it logs information

A single instance of syslogd can process both local and remote messages, if so desired.

1.1.1 What is syslogd

Many applications included in z/OS Communications Server, other IBM applications, and third-party applications that log messages use syslogd. To organize and keep track of these messages, it is important to configure syslogd.

The syslogd accepts messages from applications or another syslogd on another host and writes the messages to the proper files (on this host) or to the syslogd on yet another host, as directed by the syslogd configuration file.

1.1.2 How syslogd works

You can start the syslogd through MVS JCL or through a UNIX shell command. You can stop it using the MODIFY (F) command from the MVS console or the UNIX **kill** command to invoke the UNIX SIGTERM signal. When running, the daemon stores its process ID in `/etc/syslog.pid` or `/etc/syslog_net.pid` (network-only mode).

When the daemon completes initialization, a message is written to the MVS console indicating the successful start.

The syslogd uses UDP packets for data transfer with the well-known port number 514. *RFC 3164* describes the syslogd protocol.

The syslogd can produce large amounts of output. To prevent a bottleneck during the processing of messages that are necessary for logging, audit trails, and problem determination, syslogd is a *multithreaded* application. Log messages for each unique destination are written on a separate thread.

1.1.3 How can syslogd be deployed

The syslogd receives messages from other systems through the UDP protocol and can, therefore, act as a central repository of messages for a number of systems. However, using syslogd as a central repository for multiple systems is somewhat rare. More often, syslogd is run on each individual system and logs messages for applications on that particular system only. However, the use of Linux on System z can change this usage when Linux applications are closely related to z/OS applications. This might occur with functions such as middle-tier servers such as web servers, application servers, or Communications Controller for Linux (CCL). In these situations, it can be desirable to integrate syslogd messages from these Linux images into the z/OS environment.

A number of implementation possibilities exist for syslogd logging schemes, such as:

- Sending all messages to a single file
- Sending messages to different files based on job name
- Sending messages to different files and to a single file
- Starting two instances of syslogd

Sending all messages to a single file

The most basic use of syslogd sends all messages to a single file. The file name contains the date and a new file is automatically created daily.

Prior to starting, syslogd requires an `/etc/syslog.conf` configuration file containing at least a single logging directive. To tell syslogd to switch to a new file name, create a cron job that sends a signal to syslogd informing it that a new day has started and that it should reread the configuration file.

You can set up this configuration in a matter of minutes, and it is easy to understand. However, sending all messages to a single file makes it difficult to follow what is occurring in a single application over many hours. In addition, sending all messages to a single file does not exploit the multithreaded design of the syslogd, because each individual destination is assigned a thread. With a single log file, all message processing occurs on a single thread.

Sending messages to different files based on job name

In this case, a syslogd configuration file is created where each application sends messages to a different syslogd file. Each file is stored in a directory that contains the date in the name of the directory. Every day, the files shift to a new directory.

Prior to starting, syslogd requires an `/etc/syslog.conf` configuration file containing multiple logging directives, one per application.

This configuration can make it easy to find messages related to a particular application and follow the flow of events. This configuration also employs *syslogd isolation* by only allowing particular applications to write to syslogd facilities that should be reserved for system applications. However, this configuration makes it difficult to determine what is occurring system wide across all applications at a given point in time, and requires each job name to be specified in the `syslog.conf` file. Any job name not explicitly coded will not be saved by syslogd.

Sending messages to different files and to a single file

Prior to starting, the syslogd requires an `/etc/syslog.conf` configuration file that contains multiple logging directives—one directive per application and one additional directive that identifies the file for logging all messages into a single location.

Because some applications can generate a large volume of output, it is possible for the UNIX file systems that contains the UNIX files to fill up quickly. Semi-automatic archival mechanisms already exist for syslogd. Some installations use their own REXX or UNIX shell scripts to archive logs, swap out older log files for fresh log files, and then send a signal to the syslogd so that the daemon rereads the configuration file and continues writing log records to a new set of log files. Other installations rely on a script that is provided by IBM (`/usr/lpp/tcpip/samples/rmoldlogs`) to perform the same tasks. This script depends on the UNIX file names as defined in the `syslog.conf` rules to contain symbols for the date.

With either type of file management script, UNIX relies on an external process—the CRON daemon—to send a SIGHUP signal to syslogd once per day. This signal causes syslogd to close its UNIX files and open new files in the appropriate format. If the signal is sent just after midnight, the new file is created if the file does not already exist. The use of CRON is documented in *z/OS UNIX System Services Planning*, GA22-7800.

The syslogd introduces an automatic archival function, which is described in 1.4.2, “Description of syslogd automatic archival” on page 19.

Starting two instances of syslogd

One of the two instances receives log messages locally, and the other receives log messages from other servers (such as Linux on System z) over the network.

Prior to starting the instances either create a separate configuration file for each instance of syslogd or configure `/etc/syslog.conf` so it can be shared by both instances. The `/etc/syslog.conf` file contains multiple logging directives (one per application). It also contains `/dev/operlog` as a logging directive (one per host name or IP address), and one additional directive identifying the file that will log all specific network messages in a single location.

Note: You can start a maximum of two instances of syslogd. If you start more than one instance of syslogd on the same z/OS image, you must start one instance in local-only mode and one instance in network-only mode.

Never run just one instance of syslogd in network-only mode. If an instance of syslogd is not processing local system and application messages, then these messages are written to the MVS console and might result in message flooding on the MVS console.

1.2 Log messages to different files and to a single file

In this section, we discuss the syslogd configuration that logs to different files based on job name and also logs all messages to a single common file. To minimize DASD usage, we also discuss management techniques to prevent the z/OS UNIX file system from filling with outdated log files. This section includes the following topics:

- Description of logging to multiple files and to a single file
- Configuration of multiple files and a single file
- Verification of multiple files and a single file

1.2.1 Description of logging to multiple files and to a single file

This configuration makes it easy to find messages related to a particular application and follow the flow of events. It also provides complete information of what is occurring on the system at a given point in time. This configuration consumes more DASD than other scenarios. However, the DASD considerations can be eliminated by automated management of the log files generated by syslogd. The configuration also requires that each job name be specified, but it is ensured that messages from a job name not explicitly coded in the configuration will end up in the file that is logging all messages.

1.2.2 Configuration of multiple files and a single file

Perform the following configuration tasks:

- Plan your configuration
- Create an /etc/syslog.conf configuration file
- Create a crontab entry for the CRON daemon and syslogd
- Start syslogd
- Use a syslogd environment variable file

Plan your configuration

Before you begin, it is helpful to become familiar with and understand how to use the syslogd configuration parameters that we describe in this section.

The syslogd facilities

The syslogd uses the concept of *facility names* to group messages together. Table 1-1 lists the facility names that are supported and predefined in the syslogd implementation.

Table 1-1 Supported facilities names for syslogd

Facility name	Definition
user	Message generated by a process (user).
mail	Message generated by mail system.
news	Message generated by news system.
uucp	Message generated by UUCP system.
daemon	Generally used by server processes. The FTPD server, the RSHD server, the REXECD server, the SNMP agent, and the SNMP subagent use this facility name to log trace messages.
auth/authpriv	Message generated by authorization daemon.
cron	Message generated by the clock daemon.
lpr	Message generated by the (z/OS UNIX **lp** command) print client.
local0-7	Names for local use. The z/OS UNIX Telnet server uses the local1 facility name for its log messages.
mark	Used for logging MARK messages.
kernel	z/OS does not generate any log messages with the kernel facility, and it does not accept log messages from local applications with the kernel facility. However, syslogd on z/OS is capable of receiving log messages over the network from other syslog daemons using the kernel facility. The kernel facility can be used in rules to direct these log messages to specific destinations.

Table 1-2 lists the facilities that z/OS Communications Server uses.

Table 1-2 z/OS syslogd facilities

Application	syslogd record identification	Primary syslogd facility	Other syslogd facility
Application Transparent Transport Layer Security (AT-TLS)	TTLS	daemon	auth
Automated domain name registration (ADNR)	adnr	daemon	None
Defense Manager daemon (DMD)	DMD	local4	None
Communications Server SMTP (CSSMTP)	CSSMTP	mail	None
DHCP server	dhcpsd	user	None
FTP server	ftpd, ftps	daemon	None
IKE daemon	IKED	local4	None
NAMED	named	daemo	None

Application	syslogd record identification	Primary syslogd facility	Other syslogd facility
Network security services (NSS) server	NSSD	local4	None
Network SLAPM2 subagent	NSLAPM2	deamon	None
OMPROUTE	omproute	user	None
OPORTMAP server	oportmap	daemon	None
OREXECD	rexecd	deamon	auth
ORSHD	rshd	daemon	auth
OTELNETD	telnetd	local1	auth
Policy Agent	Pagent	daemon	None
POPPER	popper	mail	None
PWCHANGE command	pwchange	daemon	None
PWTOKEY command	pwtokey	daemon	None
rpcbind	rpcbind	daemon	None
SENDMAIL	sendmail	daemon	None
Simple Network Time Protocol daemon	sntpd	daemon	None
SNMP agent OSNMPD	snmpagent	deamon	None
syslogd	syslogd	daemon	None
TCP/IP subagent	M2SubA	daemon	None
TFTP server	tftpd	user	None
TIMED daemon	timed	user	None
TN3270E Telnet Subagent	TNSubA	daemon	None
Traffic Regulation Management Daemon (TRMD)	TRMD	daemon (used for IDS logging)	local4 (used for IPSEC logging and defensive filter logging)
Trap Forwarder daemon	trapfwd	daemon	None
z/OS Load Balancing Advisor	lbadv	daemon	None
z/OS Load Balancing Agent	lbagent	daemon	None

The syslogd configuration file

The syslogd processing is controlled by a configuration file (usually named `/etc/syslog.conf`). A sample of the latest version of syslogd configuration file is included in the `/usr/lpp/tcpip/samples/` directory with the name of `syslog.conf`. The default code page for editing syslogd is IBM-1047, but you can override this default to permit editing in a finite set of single-byte EBCDIC code pages. See *z/OS Communications Server: IP Configuration Guide*, SC31-8775 or *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for details about supported code pages.

The syslogd configuration file allows you to separate messages based on the user ID of the application generating the message, job name of the application generating the message, facility used by the application, and priority of the message generated by the application. The exploitation of this feature is called *syslogd isolation*.

Log messages can be collected from numerous network sources, including Linux hosts and networking equipment such routers, and can be filtered to write to the desired destination based on the source IP address (including subnetwork or the host name). If you specify

/dev/operlog as a directive in the configuration file, log messages can be written to the MVS operations log, which provides better performance than writing to /dev/console.

The syslogd parameters

The syslogd has many options that can be specified on the command line. These options are described in detail in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. The following options are Important:

- Create log files and directories automatically (**-c** option)

  We recommend the use of the **-c** option. With the **-c** option, syslogd can create directories and files as needed. By default, all files and directories must be created in advance. Some of the advanced file naming features found in syslogd require the use of the **-c** option so that the files can be created as needed.

- Configuration file name, which overrides file name specified in an environment file (**-f** option)

- Set permission bits for UNIX directories and files that are created automatically (the **-D** and **-F** options)

  The **-D** and **-F** parameter values are specified as an octal number of 1 to 4 characters in length. Leading zeros can be omitted. The **-D** start option specifies the default access permissions that syslogd uses when creating directories. The **-F** start option specifies the default access permissions that syslogd uses when creating log files.

  For complete details about restricted bit combinations, see the syslogd sections in *z/OS Communications Server: IP Configuration Guide*, SC31-8775 and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

- Local-only mode (**-i** option)

  The **-i** option prevents syslogd from accepting UDP messages from the network. If you do not intend to collect messages from other systems, then a UDP socket is not required and you do not specify **-i**.

- Network-only mode (**-n** option)

  The **-n** option allows syslogd to receive messages only over the network.

- Disable host name resolution (**-x** option)

  The **-x** option causes syslogd to avoid resolver calls for converting IP addresses to host names, which improves performance by avoiding IP address-to-host name resolution for network log messages.

- User ID and job name (**-u** option)

  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 (**-u** option).

- Debug mode (**-d** option)

  To run syslogd in debugging mode, use the **-d** option.

  Rule: Do not use the **-d** option for normal operations, because the default logging level produces large amounts of output. To control the amount of output if you specify this option at startup, code a value for the SYSLOGD_DEBUG_LEVEL variable in the syslogd environment variable file.

Syslogd isolation

Syslogd isolation refers to a configuration of syslogd that allows only certain job names to write log messages to particular syslogd destinations. The syslogd isolation feature is

enabled automatically when job names are included as a filter option in the syslogd configuration file rules. When you exploit syslogd isolation, you also exploit multithreading for syslogd. Recall that syslogd processes individual destinations on separate threads. (If you used one large syslogd destination file, you exploit only a single thread.)

Create an /etc/syslog.conf configuration file

The *IBM z/OS Configuration Assistant*, described in *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999, produces as part of its output, a `syslog.conf` file that includes entries for the various policies that you might have created with the Configuration Assistant. You can choose to use the output as your `syslog.conf` file, or you can use the output to enhance an existing `syslog.conf` file that you have customized for other purposes.

If you have not yet started working with Configuration Assistant, you might prefer to copy the `/usr/lpp/tcpip/samples/syslog.conf` file to `/etc/syslog.conf` as a starting point. For Example 1-1 to work, you do not need to copy the sample configuration file. However, the sample configuration file includes a large number of comments at the beginning of the file that explain the syntax of the file. If you choose to copy the sample, delete the last four lines of the configuration file, starting from the following line:

```
THIS EXAMPLE STATEMENT IS UNCOMMENTED
```

Then, add the lines shown in Example 1-1.

Example 1-1 Our /etc/syslog.conf configuration file

```
*.TRMD*.*.*     /var/syslog/%Y/%m/%d/trmd.log -F 640 -D 770
*.OMP*.*.*      /var/syslog/%Y/%m/%d/omp.log -F 640 -D 770
*.OMP*.*.err    /var/syslog/%Y/%m/%d/omp.err -F 640 -D 770
*.OMP*.*.debug  /var/syslog/%Y/%m/%d/omp.debug -F 640 -D 770
*.INETD*.*.*    /var/syslog/%Y/%m/%d/inetd.log -F 640 -D 770
*.OSNMP*.*.*    /var/syslog/%Y/%m/%d/inetd.log -F 640 -D 770
*.PAGENT*.*.*   /var/syslog/%Y/%m/%d/pagent.log -F 640 -D 770
*.FTPD*.*.*     /var/syslog/%Y/%m/%d/ftpdb.log -F 640 -D 770
*.IKE*.*.*      /var/syslog/%Y/%m/%d/inetd.log -F 640 -D 770
*.TN3270*.*.*   /var/syslog/%Y/%m/%d/tn3270b.log -F 640 -D 770
*.SYSLOGD*.*.*  /var/syslog/%Y/%m/%d/syslogd.log -F 640 -D 770
*.SENDMAIL.*.*  /var/syslog/%Y/%m/%d/sendmail.log -F 640 -D 770
*.NAMED*.*.*    /var/syslog/%Y/%m/%d/named.log -F 640 -D 770
```

If the destination is a file, it can be optionally followed by the two options, **`-F`** and **`-D`**. These parameters, described in “The syslogd parameters” on page 8, are used to specify the access permission level when a new directory and file are created.

Create a crontab entry for the CRON daemon and syslogd

You can manage syslogd log files and reinitialization with one of the following methods:

- Implement a CRON daemon.
- Use the automated archival function described in “Description of syslogd automatic archival” on page 19.

> **Important:** CRON must be configured and running to run these scripts automatically. You can find more information about CRON in *z/OS UNIX System Services Planning*, GA22-7800.

Use the following tasks to create a crontab entry that the CRON daemon can use to manage syslogd:

1. Issue the OMVS command from the TSO Ready prompt.
2. From the z/OS UNIX shell, issue the **export EDITOR=oedit** command, and then issue the **crontab -e** command.
3. Using the editor, add the lines shown in Example 1-2. Save the file, exit the editor, and issue the exit command to return to TSO.

Example 1-2 The crontab entries

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

Start syslogd

Syslogd must be started by a *superuser*. Most processes require that syslogd be running prior to the initialization of other processes. Therefore, we suggest that you start syslogd before you initialize any TCP/IP stacks. You can start syslogd using one of the following methods:

- Start syslogd from the UNIX shell command line.

  This type of initialization uses the entire shell process and does not allow you to resume other work in that shell. You must terminate the startup command with a final ampersand (*&*).

  Migration note: In prior releases, syslogd executed a UNIX `fork()` into a separate child address space, and the terminating ampersand sent the syslogd process to the background, which allowed you to resume work in the shell. The current release eliminates the UNIX `fork()` to allow APF authorizations to permit additional functions. A `fork()` removes APF authorization from the child process.

  As a result of this change, the trailing ampersand no longer frees the shell process. To escape from the shell without terminating the running syslogd, you must invoke PF2 for the OMVS subcommand prompt and enter **quit**, which returns you to MVS.

- Start syslogd from the `/etc/rc` file in UNIX.

  You can insert the syslogd startup command into the `/etc/rc` file. The lines depicted in Example 1-3 cause syslogd to start every time z/OS UNIX initializes.

Example 1-3 The /etc/rc entry to start syslogd with new job name and no environment variables

```
export _BPX_JOBNAME='syslogd'
/usr/sbin/syslogd -c -i -u -D 0755 -F 0664 & 1
```

Note: If you start syslogd using the z/OS UNIX shell command (such as from a user session or particularly from `/etc/rc`), you must add an ampersand to the end of the command to allow the command to run in the background, as shown at **1** in Example 1-3. This is not a concern if you start syslogd from a cataloged procedure.

If you choose to deploy environment variables with your syslogd, the /etc/rc script must include UNIX exports of those variables, as shown in Example 1-4.

Example 1-4 The /etc/rc entries including environment variables

```
export _BPX_JOBNAME="SYSLOGD"
export SYSLOGD_CODEPAGE="IBM-1047"
export SYSLOGD_DEBUG_LEVEL="91"
export SYSLOGD_CONFIG_FILE="//'SVT390.TCPIP.SYSLOGD.CONF(DFLTARCH)'"
/usr/sbin/syslogd -f /etc/syslog.conf -c -i -u -D 0755 -F 0664 &
```

The variables in Example 1-4 are described in “Use a syslogd environment variable file” on page 12.

If you need to enable debug mode when you initialize syslogd from the /etc/rc file or from the UNIX shell command line, you must include the location of the output file. In Example 1-5, we piped the debug output into a file named sysout.out.

Example 1-5 Syslogd startup with debug sending output to a UNIX file

```
/usr/sbin/syslogd -f /etc/syslog.conf  -c -i -u -d > /var/syslog/sysout.out &
```

- Start syslogd with JCL from MVS.

  Example 1-6 shows how to use JCL to start the syslogd. We derived this JCL example from the sample in *hlq*.SEZAINST(SYSLOGD). (On our system the sample JCL member is located in TCPIP.SEZAINST.)

Example 1-6 Started task for SYSLOGD

```
//SYSLOGD PROC
//SYSLOGD  EXEC PGM=SYSLOGD,REGION=30M,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)',
//      '/-c -i -u -f /etc/syslog.conf')  1
//SYSIN    DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSERR   DD PATH='/var/syslog/syserr',PATHOPTS=(OWRONLY,OCREAT) 2
//SYSOUT   DD PATH='/var/syslog/sysout',PATHOPTS=(OWRONLY,OCREAT) 3
//CEEDUMP  DD SYSOUT=*
//SYSIN DD DUMMY
```

In Example 1-6, we used several options 1, including the **-c** option that defines the location of the syslogd configuration file. The lines labelled 2 and 3 show that we identified output locations for debugging information, if we decide to start syslogd with the debug option (**-d**). The *PATHOPTS* parameter on the statement permits the UNIX files to be created if they are not yet built. You might want to route debugging to SYSOUT=* unless you have a special reason to create an output file.

Important: You can use the _CEE_ENVFILE environment variable in the PARM field of the JCL to point to a file that contains other environment variables. The file can be an HFS, a zFS, or a z/OS MVS data set. When it is an MVS data set, the data set must be allocated with RECFM=V.

Do *not* use RECFM=F, because this setting allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value can cause a file-not-found condition because the padded blanks are considered part of the name of the file in z/OS UNIX. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Example 1-7 depicts a variation of the JCL. At 4 we include an environment variable as a parameter, which points to an environment file at 5 that contains entries to influence the syslogd process.

Example 1-7 Started task for syslogd (includes DD for STDENV)

```
//SYSLOGD  PROC STDENV=SYSLENV
//SYSLOGD  EXEC PGM=SYSLOGD,REGION=30M,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)',
//      'ENVAR("_CEE_ENVFILE=DD:STDENV",',     4
//      '"TZ=EST5EDT"),',
//      '/-c -i -u -f /etc/syslog.conf')
//STDENV   DD DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV),DISP=SHR     5
//SYSPRINT DD SYSOUT=*
//SYSERR   DD PATH='/var/syslog/syserr',PATHOPTS=(OWRONLY,OCREAT)
//SYSOUT   DD PATH='/var/syslog/sysout',PATHOPTS=(OWRONLY,OCREAT)
//CEEDUMP  DD SYSOUT=*
//SYSIN DD DUMMY
```

Use a syslogd environment variable file

Certain command line start options can use very large values, for example UNIX file names. It is difficult to specify these parameters in a cataloged procedure because of the 100-character limit for the PARM parameter on the JCL EXEC statement. Example 1-7 points to an environment variable file member that allows us to specify variables without the restrictions of a PARM parameter on the JCL. You can include numerous entries in this file. In Example 1-8, we coded several of these environment variables.

Example 1-8 Environment variables for SYSLOGD: TCPIP.SC31.STDENV(SYSLENV)

```
SYSLOGD_CODEPAGE=IBM-1047             1
SYSLOGD_CONFIG_FILE=/etc/syslog.conf  2
SYSLOGD_DEBUG_LEVEL=127               3
SYSLOGD_PATH_NAME=/dev/log            4
```

In this example, the numbers correspond to the following information:

1. `SYSLOGD_CODEPAGE`

Used by the syslogd to specify the EBCDIC code page to be used for the configuration file. The default code page is `IBM-1047`. If desired, you can use quotation marks to enclose the code page name, for example “IBM-1047”. The other supported code pages are documented in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Some of the new automatic archival configuration statements can contain POSIX variant characters or other special characters. This requires support for code pages other than IBM-1047, so the configuration file can be edited and viewed in the native code page.

2. `SYSLOGD_CONFIG_FILE`

Specifies the name of the syslogd configuration file. This value is overridden by the *-f* option in the syslogd startup.

3. `SYSLOGD_DEBUG_LEVEL`

Specifies the debug level to be used by syslogd. The default debug level is 127, which includes all debug information. You use this value only if you use the **-d** start option for syslogd. If you encounter problems in syslogd that are related to multiple threads or

locking, make sure the debug level includes the values 32 and 64 when collecting documentation.

A useful debug level to use for many types of debugging and when collecting documentation is 91. This debug level excludes debugging information that gets written for each log message sent to syslogd. However, you might need to use the default debug level of 127 for certain types of problems.

4. SYSLOGD_PATH_NAME

Specifies the path name for the datagram socket. This value is overridden by the **-p** option in the syslogd startup. However, you do not have to use this variable at all. We include it here only for the sake of completeness.

1.2.3 Verification of multiple files and a single file

To verify that syslogd is working correctly, go to the directory that contains the log files and check each file. Example 1-9 shows the output of **/bin/ls** that was issued while in our /var/syslog/2010/09/29/ directory.

Example 1-9 Output of /bin/ls in our log directory

```
# cd /var/syslog/2010/09/30
CS02 @ SC31:/SC31/var/syslog/2010/09/30>ls -l
total 144
-rw-r-----   1 SYSPROG  SYS1       62076 Sep 30 16:45 ftpdb.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 inetd.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 named.log
-rw-r-----   1 SYSPROG  SYS1        3670 Sep 30 16:47 netlog.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 omp.debug
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 omp.err
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 omp.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 pagent.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 sendmail.log
-rw-r-----   1 SYSPROG  SYS1         532 Sep 30 15:47 syslogd.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 tn3270b.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 trmd.log
```

Example 1-10 shows the contents of one of our log files.

Example 1-10 Contents of the OMP.log file

```
Sep 30 16:52:16 WTSC31/TCPIP    OMPB     omproute[67240087]: EZZ7800I OMPB starting
Sep 30 16:52:16 WTSC31/TCPIP    OMPB     omproute[67240087]: EZZ7845I Established affinity
with TCPIPB
Sep 30 16:52:16 WTSC31/TCPIP    OMPB     omproute[67240087]: EZZ7817I Using default OSPF
protocol 89
Sep 30 16:52:22 WTSC31/TCPIP    OMPB     omproute[67240087]: EZZ7817I Using default OSPF
protocol 89
Sep 30 16:52:22 WTSC31/TCPIP    OMPB     omproute[67240087]: EZZ7838I Using configuration
file: 'TCPIPB.TCPPARMS(OMPB31)'
Sep 30 16:52:22 WTSC31/TCPIP    OMPB     omproute[67240087]: EZZ7883I Processing interface
from stack, address 10.1.2.21, name OSA2080I, index 1, flags 441
Sep 30 16:52:22 WTSC31/TCPIP    OMPB     omproute[67240087]: EZZ7883I Processing interface
from stack, address 10.1.2.22, name OSA20A0I, index 2, flags 441
```

1.3 Starting two syslogd instances

In this section, we discuss setting up and running of two instances of syslogd. One instance is in local mode for logging message for the local system, and the other instance is in network mode for logging messages that are received from remote systems over the network.

This section includes the following topics:

- Description of two syslogd instances
- Configuring two syslogd instances
- Verification for running two syslogd instances

1.3.1 Description of two syslogd instances

Running two instances of syslogd helps ensure that local syslogd logging is not adversely affected by the number of remote messages being forwarded to z/OS. These configurations make it easy to find messages related to a particular application and follow the flow of events. They also provide complete information about what is occurring on the system and other hosts at a given point in time.

However, because network syslogd messages are delivered through UDP, delivery of messages is not guaranteed. Moreover, messages cannot be delivered to z/OS under some conditions. The use of IPSEC should be considered for protecting the syslogd's UDP port 514 when running in local or network-only mode. For more information, see *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7801.

Note: A maximum of two instances of syslogd can be started. However, if you are going to start more than one instance of syslogd on the same z/OS image, then one instance must be started in local-only mode and one instance must be started in network-only mode.

Never run just one instance of syslogd in network-only mode. If an instance of syslogd is not processing local system and application messages, these messages are written to the MVS console and might result in message flooding on the MVS console.

1.3.2 Configuring two syslogd instances

Complete the following tasks to configure two syslogd instances:

1. Plan for syslogd remote logging.
2. Create `/etc/syslog.conf` configuration files.
3. Create crontab entries for syslogd.
4. Start syslogd.

Planning for syslogd remote logging

The syslogd application can act as a message receiver, receiving syslogd messages from remote syslogd implementations. A syslogd is configured as network only mode (**`-n`** option start parameter). It can also send messages to a remote syslogd if it is configured for local mode (with the **`-i`** option start parameter).

Proper planning is mandatory to ensure that the additional remote message traffic that syslogd receives does not create performance issues or impede the ability for the local z/OS syslogd to perform its processing.

If you anticipate a high syslogd network message rate, consider configuring one network-only instance and one local-only instance of syslogd.

Example 1-11 depicts hosts for which syslogd processes incoming syslogd messages. After initialization, syslogd begins receiving remote messages. It first determines whether the incoming messages match any of the specified configuration statements. In this example, syslogd processes only the remote messages that have a source IP address associated with the host wtc30 and network 10.42.105.0 and then logs these messages in the MVS operations log. Only critical messages from host 10.43.110.15 are logged into `/tmp/otherlog`.

Example 1-11 Identifying received remote syslogd messages

```
(wtc30).*.*              /dev/operlog
(10.42.105/24).*.*       /dev/operlog
(10.43.110.15).*.crit    /tmp/otherlog
```

Example 1-12 shows an entry in the configuration file that writes all messages with a priority of `crit` and higher to the syslogd on host 192.168.1.9. The SYSLOGD in host 192.168.1.9 must be activated in network mode.

Example 1-12 Sending messages to remote host

```
*.crit      @192.168.1.9
```

Tip: The parentheses in Example 1-11 denote that it is a remote host. The host can be specified by IPv4 address, by IPv6 address, or by a name that resolves to an IPv4 or IPv6 address. The at sign (@) in Example 1-12 denotes that the host is a remote host.

Creating /etc/syslog.conf configuration files

In our testing, we established only one configuration file to be shared by the two instances of syslogd. Example 1-13 shows our configuration file.

Tip: If you intend to use `*.*` as a filter, then configure two separate configuration files (one for each syslogd) because the wildcard `*.*` matches *all* host names and IP addresses.

In this example, the network-only mode syslogd does not send its log messages to another syslogd. Therefore, define `syslogd*` as the job name in a configuration file of network-only mode syslogd if you want to collect log messages of the network-only mode syslogd also.

Example 1-13 The /etc/syslog.conf configuration file for shared local and network syslogd instances

```
######################################################################
#   For local-only mode                                              #
######################################################################
*.TRMD*.*.*     /var/syslog/%Y/%m/%d/trmd.log -F 640 -D 770
*.OMP*.*.*      /var/syslog/%Y/%m/%d/omp.log -F 640 -D 770
*.OMP*.*.err    /var/syslog/%Y/%m/%d/omp.err -F 640 -D 770
*.OMP*.*.debug  /var/syslog/%Y/%m/%d/omp.debug -F 640 -D 770
*.INETD*.*.*    /var/syslog/%Y/%m/%d/inetd.log -F 640 -D 770
*.OSNMP*.*.*    /var/syslog/%Y/%m/%d/inetd.log -F 640 -D 770
*.PAGENT*.*.*   /var/syslog/%Y/%m/%d/pagent.log -F 640 -D 770
*.FTPD*.*.*     /var/syslog/%Y/%m/%d/ftpdb.log -F 640 -D 770
*.IKE*.*.*      /var/syslog/%Y/%m/%d/inetd.log -F 640 -D 770
*.TN3270*.*.*   /var/syslog/%Y/%m/%d/tn3270b.log -F 640 -D 770
*.SYSLOGD*.*.*  /var/syslog/%Y/%m/%d/syslogd.log -F 640 -D 770
```

```
*.SENDMAIL.*.*  /var/syslog/%Y/%m/%d/sendmail.log -F 640 -D 770
*.NAMED*.*.*    /var/syslog/%Y/%m/%d/named.log -F 640 -D 770
#####################################################################
#  For network-only mode                                            #
#####################################################################
# Write all messages with priority err and higher that arrive from
# host WTSC31 to the netlog file.
#####################################################################
 (WTCS31).*.err    1 /var/syslog/%Y/%m/%d/netlog.log -F 640 -D 770
#
```

In Example 1-13 on page 15, note the following point:

1. Receive from remote host `WTCS31` all messages with priority err and higher and write them in the file located at `/var/syslog/%Y/%m/%d/netlog.log`.

You can specify either the IP address or a host name.

To be able to send messages, the SYSLOGD in the remote server has to be started with option *-n* and must have an entry defining where to send its messages, as shown at indicator **1** of Example 1-14.

Example 1-14 SYSLOGD configuration on the remote server

```
#####################################################################
#  For network-only mode                                            #
#####################################################################################
# Send all messages with priority err and higher to the syslogd on
# host WTCS31.
#####################################################################
 *.err      @WTCS31 1
#
```

Creating crontab entries for the CRON daemon and syslogd

You can manage syslogd log files and reinstallation using one of the following methods:

- Implement a CRON daemon.
- Use the automated archival function described in “Description of syslogd automatic archival” on page 19.

If you are going to manage syslogd archiving and restarts with the CRON daemon, you need to create crontab entries for the CRON daemon. To create a crontab entry that the CRON daemon can use to manage syslogd complete the following steps:

1. Issue the command OMVS from the TSO Ready prompt.
2. From the z/OS UNIX shell, issue the command **`export EDITOR=oedit`** and then issue the command **`crontab -e`**.
3. Using the editor, add the two lines shown in Example 1-15.
4. Save the file and exit the editor. These entries allow log files to be re-created daily.

Example 1-15 The crontab entries

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

When syslogd is started in the network-only mode, the syslogd stores its process ID in the `/etc/syslog_net.pid` in addition to the `/etc/syslog.pid` file.

Starting syslogd

We issued the z/OS UNIX shell commands shown in Example 1-16.

Example 1-16 Starting syslogd

```
export _BPX_JOBNAME='syslogd'
/usr/sbin/syslogd -f /etc/syslog.conf -c -i -D 0755 -F 0664 &
/usr/sbin/syslogd -f /etc/syslog.conf -c -n -x &
```

To start syslogd every time z/OS UNIX starts, add the same lines to `/etc/rc`, as shown in Example 1-17.

Example 1-17 The /etc/rc entries to start syslogd

```
export _BPX_JOBNAME='syslogd'
/usr/sbin/syslogd -f /etc/syslog.conf -c -i -D 0755 -F 0664 &
/usr/sbin/syslogd -f /etc/syslog.conf -c -n -x &
```

1.3.3 Verification for running two syslogd instances

To verify that the syslog daemons are working correctly, you can use a **`ps -ef | grep syslogd`** command to verify that the log files in the directory match the defined configuration files. Example 1-18 shows a process list.

Example 1-18 Process list

```
CSO2 @ SC31:/u/cs02>ps -ef | grep syslogd
 SYSPROG   33685686          1  - 13:46:23 ttyp0000  0:00 /usr/sbin/syslogd -c -i
-u -f /etc/syslog.conf 1
 SYSPROG     131255          1  - 13:46:23 ttyp0000  0:00 /usr/sbin/syslogd -c -n
-u -f /etc/syslog.conf 2
```

In Example 1-18, note the following points:

1. Shows the local-only mode syslogd.
2. Shows the network-only mode syslogd.

Example 1-19 shows the output of an **`ls`** command that was issued while in the `var/syslog/2010/09/30` directory.

Example 1-19 Output of the ls command

```
CSO2 @ SC31:/u/cs02>cd /var/syslog/2010/09/30
CSO2 @ SC31:/SC31/var/syslog/2010/09/30>ls -l
total 32
-rw-r-----   1 SYSPROG  SYS1        4407 Sep 30 15:54 ftpdb.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 inetd.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 named.log
-rw-r-----   1 SYSPROG  SYS1         740 Sep 30 15:55 netlog.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 omp.debug
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 omp.err
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 omp.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 pagent.log
```

```
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 sendmail.log
-rw-r-----   1 SYSPROG  SYS1         532 Sep 30 15:47 syslogd.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 tn3270b.log
-rw-r-----   1 SYSPROG  SYS1           0 Sep 30 15:47 trmd.log
```

Example 1-20 shows the output of `var/syslog/2010/09/30/netlog.log` on SC31 (also known as *wtsc31*). The log entries (`wtsc30.itso.ibm.com`) show that those messages were received from a remote host.

Example 1-20 The contents of `netlog.log` *on SC31*

```
Sep 30 15:50:22 wtsc30.itso.ibm.com/N/A      N/A      TCPIP    OMPA
omproute[50397366]: EZZ8107I OMPA subagent: Connection to SNMP agent Dropped
Sep 30 15:50:53 wtsc30.itso.ibm.com/N/A      N/A      TCPIP    OMPA
omproute[50397366]: EZZ8108I OMPA subagent: reconnected to SNMP agent
Sep 30 15:51:22 wtsc30.itso.ibm.com/N/A      N/A      CSO2     CSO24
fomtlout[65585]: FSUM2301 The end of the session was not recorded.  An unexpected
error occurred.  Error code = 214, return value = 0, errno = 0 (X'00000000'),
reason code = 00000000, message = 'EDC5000I No error occurred.'
Sep 30 15:55:32 wtsc30.itso.ibm.com/N/A      N/A      TCPIP    OMPA
omproute[50397366]: EZZ8107I OMPA subagent: Connection to SNMP agent Dropped
```

Example 1-21 shows the contents of the syslogd log file.

Example 1-21 The syslogd log file

```
Sep 30 15:47:30 WTSC31/CSO2     SYSLOGD  syslogd: FSUM1220 syslogd: restart
Sep 30 15:47:30 WTSC31/CSO2     SYSLOGD  syslogd: FSUM1237 Job SYSLOGD  running in
local-only mode 1
Sep 30 15:47:30 WTSC31/CSO2     SYSLOGD  syslogd: FSUM1232 syslogd: running
non-swappable
Sep 30 15:47:30 WTSC31/CSO2     SYSLOGD  syslogd: FSUM1220 syslogd: restart
Sep 30 15:47:30 WTSC31/CSO2     SYSLOGD  syslogd: FSUM1238 Job SYSLOGD  running in
network-only mode 2
Sep 30 15:47:30 WTSC31/CSO2     SYSLOGD  syslogd: FSUM1232 syslogd: running
non-swappable
```

In Example 1-21, key log messages are as follows:

1. Shows Job SYSLOGD local instance running in local-only mode.
2. Shows Job SYSLOGD network instance running in network-only mode.

1.4 The syslogd functions

In this section, we discuss the syslogd operator commands, how to archive an active z/OS UNIX log file in an MVS data set, and the syslogd browser function. This section includes the following topics:

- The syslogd operator commands
- Description of syslogd automatic archival
- The syslogd browser and search facility

1.4.1 The syslogd operator commands

You can use the MVS console commands listed in Table 1-3 to manage the syslogd started task.

Table 1-3 Operator syslogd commands

Operator command	Function
P *procname*	STOP the syslogd
F *procname*,RESTART	RESTART the syslogd, causing the syslogd to reread its configuration file. This command is equivalent to issuing the UNIX SIGHUP signal.
F *procname,*ARCHIVE	Perform on demand syslogd ARCHIVE
F *procname*,DISPLAY,ARCHIVE	Display the current status of the automatic ARCHIVE function.

Note: In prior releases, the job name of the syslogd contained an extra character because of the UNIX forking function. This character no longer becomes part of the job name, because a `fork()` is no longer executed as part of syslogd initialization. For example, a name that was *SYSLOGD1* previously is now *SYSLOGD*. This change makes it easier to issue operator commands, because the job name is predictable. It also retains APF authorizations that are otherwise lost when the `fork()` produces a child process in a separate address space.

1.4.2 Description of syslogd automatic archival

There is a semi-automatic archival system that many installations use to manage syslogd log files. (See “Sending messages to different files and to a single file” on page 4.) Such methods use scripts and the CRON daemon. You can exploit automatic archival of syslogd log files.

The syslogd automatic archival archives the contents of a z/OS UNIX log file to an MVS data set. This function adds a fully automatic archival mechanism to syslogd, and it also supports on-demand archival if needed. The automatic archival function also reacts to “file full” conditions on the syslogd log files. The MVS archival data set can either be a sequential data set (where low-level qualifiers specify date and time) or a new generation in a generation data group (GDG). Each file destination in syslogd has an archive type attribute associated with it that is derived from the syslogd configuration file, as shown in Table 1-4.

Table 1-4 Archive type attributes for file types

Archive types	Description
NONE	No archive processing for this file
GDG	Archive done to an MVS generation data set group
SEQ	Archive done to a sequential MVS data set
CLR	The content of the file is never archived, but the file is cleared every time an archive operation is performed for the destination; this file destination is specified with the **–X** option
HFS	z/OS UNIX files based on use of the percentage symbol (%)

Note: If you use generation data group (GDG) data sets as an archive destination, the GDG base must already be in existence. The following sample JCL creates a GDG base called *USER1.SYSARCH*:

```
//USER1X JOB MSGLEVEL=(1,1),MSGCLASS=D,NOTIFY=USER1
//GDGA EXEC PGM=IDCAMS
//*
//GDGMOD DD DSN=USER1.SYSARCH,
// VOL=SER=CPDLB1,
// UNIT=SYSALLDA,
// SPACE=(TRK,(0)),
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=6800,DSORG=PS),
// DISP=(,KEEP)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE GENERATIONDATAGROUP
```

System symbols can be included in parts of the target data set names. The space requirements of the target data sets do not need to be determined; syslogd takes care of that. The syslogd retries previously failed archives automatically at the next archive event.

You can monitor console messages for failed archives to correct any problems, and syslogd will eventually archive all previously failed files successfully. You an use an operator command to display the utilization of the syslogd UNIX file systems.

Figure 1-2 shows a file archive timeline.

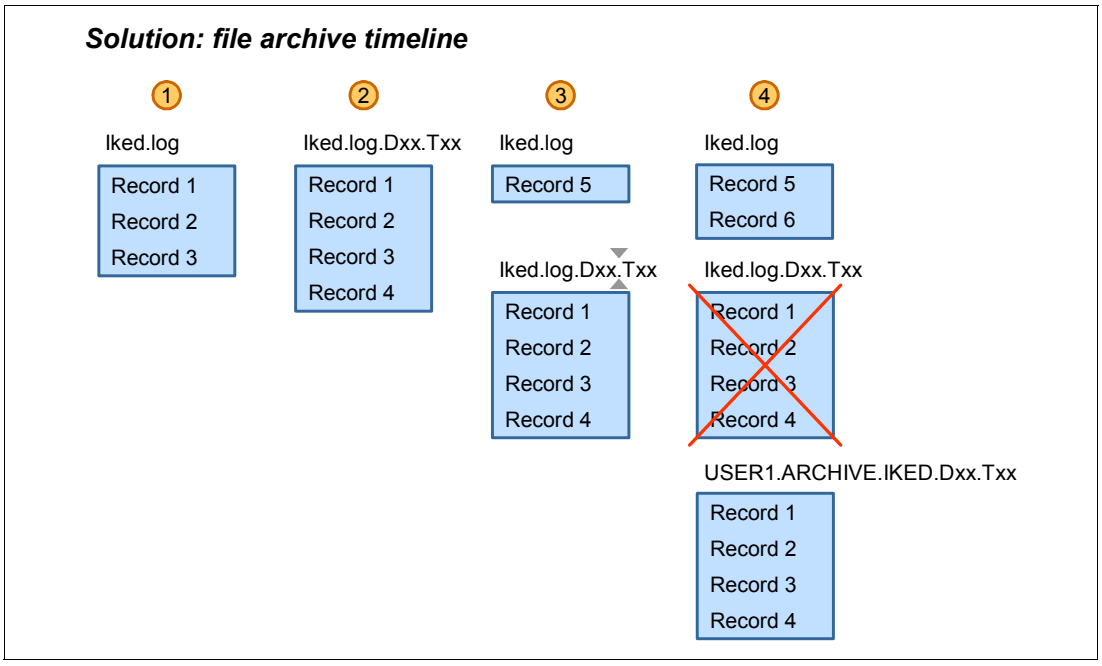


Figure 1-2 The archive timeline

The time line for an archive event for a single file follows these basic steps. In this example, we named the log file as `iked.log`.

1. The `iked.log` is open, and syslogd is writing records to it.
2. The syslogd renames the open log file with a unique date and time suffix. The file is still open so that syslogd can continue to write records to it.
3. The renamed file is closed, and the original `itso.log` file is reopened. The syslogd can continue to write to the open `iked.log` file.
4. The renamed file is archived into a target data set, and the renamed file is deleted.

You can trigger the syslogd automatic archival by using one of the following methods:

- At a specific point in time (for example, shortly after midnight).
- When the utilization of one of the file systems to which the z/OS UNIX log files are written exceeds a configurable threshold (file system percentage full).
- Archive using an operator command.

All eligible files are archived for the time of day and operator command triggers. A file system threshold trigger causes files to be archived from largest to smallest until the threshold is reached.

Important: You must specify the **-c** start option when using the automatic archival function, because syslogd must be able to create new destination files dynamically.

Tip: For the file threshold trigger, syslogd attempts to reduce the space that is used by archiving files until half of the configured threshold is reached. For example, if you configure 80% as the threshold, syslogd archives files until the file system reaches 40% utilization. A console message is issued if all eligible files are archived but the file system utilization was not able to be reduced.

Configuring syslogd for automatic archiving

Prior to configuring syslogd for automatic archiving, ensure that you have the following prerequisites in place:

- The syslogd initialization must include the **-c** start option.
- If you are using generation data group (GDG) data sets, these data set must already be in existence prior to syslogd initialization.

To configure syslogd automatic archival, complete the following steps:

1. Configure global configuration statements
2. Configure target data set parameters for groups of syslogd rules
3. Configure target data set parameters for each individual syslogd rule

Configure global configuration statements

The parameters in Table 1-5 describe the time of day and system threshold triggers.

Table 1-5 Global configuration statements

Global configuration statements	Format	Description
ArchiveTimeOfDay	hh:mm	Specifies the time of day using a 24 hour clock (no default exists)
ArchiveCheckInterval	mins	Specifies the interval for checking the system utilization (default is 10)
ArchiveThreshold percentage	Percentage	Specifies the percentage full threshold from 0 to 99 (default is 70). Specify 0 to not perform threshold based archiving

Parameter for groups of syslogd rules

The parameter for groups of syslogd rules is `BeginArchiveParms`, which specifies the target data set prefix and allocation parameters. This parameter is required for automatic archival, and it must be repeated for each group of syslogd rules. It has the following subparameters:

- DSNPrefix
- Volume
- Unit
- MgmtClass
- StorClas
- RetPd

Each specified statement applies to the rules that follows it until another instance of this statement is specified.

For more information about this parameter, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Parameters for individual syslogd rules

The default for all UNIX files is not to perform automatic archival. If you want to use this function, you must configure it explicitly using one of the following choices for each rule that contains a UNIX file log destination:

- Archive the file using the **-N** parameter.
- Reinitialize the file (delete its contents) when an archive occurs with the **-X** parameter. Use this option with care, because the contents of the file are lost
- Do nothing by not using either of these parameters.

Note: You cannot specify **-N** or **-X** with the existing **-D** or **-F** parameters.

Verification of a scenario for syslogd archiving

We implemented syslogd with an automatic archival at 01:00 each day and with an archive threshold at 80%, which separated the daemon logs from other log files. See Example 1-22.

Example 1-22 Entries for archiving in syslogd.conf

```
BeginArchiveParms
DSNPrefix CSO2.TRACE 1
Unit SYSDA
EndArchiveParms
daemon.debug /var/syslog/logs/daemon.trace -X DAEMON 2
*.debug      /var/syslog/logs/debug.trace -N  DEBUG 2
#
BeginArchiveParms
DSNPrefix CSO2.SYSLOG 1
Unit SYSDA
EndArchiveParms
ArchiveTimeOfDay 01:00 3
ArchiveThreshold 80 4
*.*;daemon.none /var/syslog/logs/syslog.log -N LOG 2
daemon.*;daemon.debug  /var/syslog/logs/syslog.log -N DAEMON 2
```

Note: You must precede the rule with a valid `BeginArchiveParms` statement that specifies the data set name prefix.

The field 1 identifies the prefix of the data set name that is created by syslogd automatic archival. `ArchiveTimeofDay` 3 sets the time of day chosen for archival and `ArchiveThreshold` 4 sets the percentage full at which to start archiving. The rules 2 that follow each `BeginArchiveParms/EndArchiveParms` set apply to that data set prefix. In the rule 4 we use the syslogd priority of *none* to separate the daemon logs from the log.

After we started the syslogd and the archive is done (either at the time of day chosen or when the file full percentage is reached), a FSUM1260 message displays, as shown in Example 1-23.

Example 1-23 Message after archive is complete

```
FSUM1260 SYSLOGD ARCHIVE COMPLETE FOR 2 FILES
```

The syslogd archiving process diagnosis

To see an overview of the syslogd archive status and file system utilization, use the MODIFY DISPLAY ARCHIVE DETAIL command, as shown in Example 1-24 on page 24. You can execute this command even if you are experiencing problems. The detailed UNIX file display is sorted from the largest file to the smallest.

Example 1-24 Modify syslogd display archive command

```
F SYSLOGD,DISPLAY,ARCHIVE,DETAIL
FSUM1268 FILE SYSTEM DETAILS 810
NAME=OMVS.SC31.VAR
PATH=/SC31/var
512-BLOCKS=   1900800 USED=     49448 AVAIL=   1851352 USAGE=  3% 1
   FILE SIZE USAGE ARCHIVE PATH
         280    0% FILE    /var/syslog/2010/09/30/ftpdb.log
          14    0% FILE    /var/syslog/2010/09/30/omp.debug
          14    0% FILE    /var/syslog/2010/09/30/omp.log
           1    0% FILE    /var/syslog/2010/09/30/syslogd.log
           1    0% FILE    /var/syslog/2010/09/30/omp.err
5 OF 14 RECORDS DISPLAYED 2
```

Take into account that the total percentage of all listed files is less than 1% but that the file system is shown as 3% full **1**. In this case, the remaining 3% of the file system utilization might consist of files that are not managed by syslogd. It is not possible to know because the default MAX value of 5 **2** is used. You can view all the files that syslogd manages if you execute the command and override the default MAX value as follows:

```
F SYSLOGDB,DISPLAY,ARCHIVE,DETAIL,MAX=*
```

Suggestion: You can dedicate HFS or zFS file systems to syslogd to get the most benefit from automatic archival based on file system utilization. Logs that reside in a temporary file system (TFS) can take advantage of archival, but the percentage of file system usage is irrelevant in a TFS scenario. As a result, automatic archival based on file system utilization is not effective.

If you receive the FSUM1259 message, check for the cause of the error. Possible causes can be a lack of space to perform the archive or a failed allocation because of a configuration error. If a failure occurs, syslogd attempts to archive the file again when the next archive trigger event occurs, but it is possible that all archive attempts will continue to fail. If all attempts fail, the original UNIX file to be archived still exists in the UNIX file system, renamed with a suffix of the form D*yymmdd*.T*hhmmss*, and the target data set might also exist and contain partial data. Examine the error messages in the syslogd destination output file for the daemon facility to determine the files that failed, and manually recover or delete such files and the associated partial archive data sets.

Example 1-25 shows that the syslogd did not have sufficient authority to create the requested file and that the archive failed.

Example 1-25 Message after archive has failed

```
ICH408I USER(OMVSKERN) GROUP(OMVSGRP ) NAME(####################) 033
  CS06.TRACE.DEBUG.D090830.T155001 CL(DATASET ) VOL(COMST1)
  DEFINE - INSUFFICIENT AUTHORITY
ICH408I USER(OMVSKERN) GROUP(OMVSGRP ) NAME(####################) 034
  CS06.SYSLOG.LOG.D090830.T155001 CL(DATASET ) VOL(COMST4)
DEFINE - INSUFFICIENT AUTHORITY
FSUM1259 SYSLOGDB ARCHIVE FAILED FOR 1 FILES
```

Note: The default is not to perform syslogd archival.

1.4.3 The syslogd browser and search facility

The syslogd log has important information, but the information is not easy to find or to view. You can configure syslogd to write to many files based on message selections using the associated message attributes, but it can still be difficult to browse for the exact information that you need in all these files.

The syslogd browser provides an easy-to-use interface to access the messages that syslogd captures on a z/OS system. With the browser, you can access archived syslogd messages when such archives are created using the syslogd archive function. Archives based on percentage symbols (%) are also supported, as long as such archive files remain in the directory in which they were originally created.

The browser provides a search mechanism that allows you to search selected active z/OS UNIX files or archives based on various search arguments. It also supplies a “guide-me” function that allows you to enter syslogd rule criteria and that guides you to the active z/OS UNIX file or files to which such messages go.

Configuring the syslogd browser

To start the syslogd browser facility, complete the following steps:

1. Configure the ISPF libraries
2. Integrate the syslogd browser into ISPF
3. Add the syslogd browser to an ISPF menu panel

Configure the ISPF libraries

Table 1-6 lists the libraries that are required in your TSO environment.

Table 1-6 Data sets for implementing the syslogd browser function

Library data set	Definition
hlq,SEZAPENU	ISPF panel library
hlq.SEZAMENU	ISPF message library
hlq.SEZAEXEC	REXX program library

You can allocate ISPF and REXX libraries using DD names in your TSO LOGON procedure or TSO LOGON CLIST. z/OS CS delivers ISPF components for panels, messages, and REXX programs, All components of the syslogd browser have member names that start with `EZASY`*`xxx`*.

Integrate the syslogd browser into ISPF

You can start the syslogd browser using one of the following methods:

- If TCPIP ISPF and REXX libraries are allocated, then start the EZASYRGO REXX program.
- If TCPIP ISPF and REXX libraries are not allocated, first copy EZABROWS to your REXX library and make local modifications, then start the EZABROWS program.

You can use the EZABROWS to start the REXX to start the browser, or you can copy EZABROWS to a REXX library that is allocated to your TSO environment.

Note: You must customize the copied EZABROWS to identify high-level qualifiers of your z/OS CS ISPF libraries.

Add the syslogd browser to an ISPF menu panel

Example 1-26 assumes that you have allocated ISPF and REXX libraries to your standard TSO setup.

Example 1-26 Entries for the ISPF menu panel

```
Opt =>
FM  FILEMAN  - File Manager
S2  SDF II   - SDF II Functions
ADM41 ADM41  - DB2 Administration Tool V41 and Object Compare V2
ADM42 ADM42  - DB2 Administration Tool V42 and Object Compare V2
ADM51 ADM51  - DB2 Administration Tool V51 and Object Compare V2
AT  DB2AUT   - DB2 Automation Tool
LA  DB2ALA   - DB2 Log Analysis Tool
LA2 DB2ALA   - DB2 Log Analysis Tool V1.2 pre-GA SMP packaging
LA3 DB2ALA   - DB2 Log Analysis Tool V1.2 GA SMP packaging +maint
LA4 DB2ALA   - DB2 Log Analysis Tool V1.2 v04FEF2002
OR  DB2AUO   - DB2 Object Restore Tool
CD  CANDLE   - CANDLE DB2 TOOLS
ESA ESA      - Electronic Service Agent
LGBR CS LGBR - syslogd browser
X   EXIT     - Terminate ISPF using list/log defaults
```

You can add the syslogd browser option to any of your ISPF menu panels, or you can invoke it from ISPF option 6 as follows:

```
EXEC 'your.REXX.library(EZABROWS)'
```

If you have not done the allocation, use the EZABROWS command instead of the following EZASYRGO command:

```
'CMD(EZASYRGO) NEWAPPL(EZAS)'
```

Files and data sets supported by the syslogd browser

The syslogd browser supports the following log files:

- The browser enables users to access the active z/OS UNIX files to which syslog is currently writing. An active syslog file is always a z/OS UNIX file, and it should be placed in a file system that is used only by syslogd. It cannot be an MVS data set.
- An archive is a z/OS UNIX file or an MVS data set to which syslogd no longer writes. It can be a UNIX file residing in the UNIX file system. The browser supports files that are named with selected percentage symbols (%), and these old files must remain the directories where they were originally created. It can be an MVS data set, created with the syslogd archive function.

Note: The browser can also use a z/OS UNIX staging file. A *staging file* is a transient file. It holds the contents of an active log file that is no longer being written to while the file is awaiting archival to an MVS data set by the syslogd archive function. If the archive to MVS data sets fails, the staging file will remain in the UNIX file system, and syslogd will retry the archive later. An example staging file name might be `/var/syslog.log.D081211.T000057`.

Verifying the syslogd browser

In a few cases, the syslogd browser prompts for the names or data set names of the log files that you want to browse. When prompted, enter the name using one of the following formats:

- Enter a z/OS UNIX file name starting with a slash, for example `/var/log/logfile`.
- Enter a fully qualified MVS data set name starting and ending with a single quotation mark (standard TSO syntax), for example `'TCPIP.LOG.LOGFILE'`.
- Enter an unqualified MVS data set name starting with any other valid symbol or alphanumeric character. The user's TSO prefix (if defined) is prefixed to the data set name. If no prefix is defined, the user ID is prefixed (standard TSO syntax) to `LOG.LOGFILE`. For example, if the user's prefix is `USER1`, then the resulting file name is `USER1.LOG.LOGFILE`.

Example 1-27 shows the syslogd browser entry panel.

Example 1-27 Syslogd browser entry panel

```
*------------------------ z/OS CS Syslogd Browser ----------- Row 1 to 1 of 1
 Command ===>                                                 Scroll ===> PAGE

 Enter syslogd browser options
   Recall migrated data sets ==> NO    1(Yes/No) Recall data sets or not
   Maximum hits to display   ==> 200   2(1-99999) Search results to display
   Maximum file archives     ==> 30     (0-400) Days to look for file archives
   Display start date/time   ==> YES   3(Yes/No) Retrieve start date/time
   Display active files only ==> NO    4(Yes/No) Active files only, no archives
   DSN Prefix override value ==>       5

 Enter file or data set name of syslogd configuration, or select one from below:

   File/DS Name ==> /etc/syslog.conf

 Press ENTER to continue, or press END PF key to exit without a selection
 Line commands: S Select, R Remove from list, B Browse content, E Edit content
 Cmd Recently used syslogd configuration file or data set name
 --- ----------------------------------------------------------------------
     /etc/syslog.conf
```

After accessing the syslogd browser ISPF panel shown in Example 1-27, you must define several options. If you specify `NO` 1, you cannot access MVS data sets that are migrated. At 2, you can specify the maximum number of hits that you want displayed as the result of a search operation. You can also set this maximum on the search panel.

If you use z/OS UNIX file archives that are based on a file naming convention that uses percentage symbols (%) for day, month, and year, the syslogd browser looks for archives in the directory in which the active z/OS UNIX file resides. The display start date and time option is used to control the display of the start date and time for each active file and each archive 3. Set this option to `NO` if you do not need it and want to improve the performance of the syslogd browser initialization.

The "Display active files only" option, shown at 4, controls whether the syslogd browser is used for browsing the currently active syslogd files only or whether it is used for browsing both active syslogd files and archives. Set this option to `NO` if you know you are browsing only the active syslogd files to improve the performance of the syslogd browser initialization.

The “DSN Prefix override value” option, shown at 5, overrides the DSNPREFIX keyword in your syslogd configuration file. This option is particularly useful if you use system symbols in your DSNPREFIX and want to browse the syslogd files of a logical partition (LPAR) other than the one to which you are logged in.

Syslogd destination rules

Example 1-28 shows the syslogd destination rules that direct messages to z/OS UNIX files. After parsing a syslogd configuration file, all z/OS UNIX file destinations are selected, and all associated available archives are located. The syslogd destination view shown in Example 1-28 is the main panel of the syslogd browser interface from which you select other functions.

Example 1-28 Syslogd destination

```
*------------------------ z/OS CS Syslogd Browser ---------- Row 1 to 3 of 17
OPTION ===>                                                 Scroll ===> PAGE

Select one of the following, or press END PF key to exit the syslogd browser

  1 Change current syslogd configuration file and/or options
  2 Guide me to a possible syslogd destination    2
  3 Clear guide-me hits (indicated by ==> in the Cmd column)
  4 Search across all active syslogd files

Current config file ==> /etc/syslog.conf

Line commands: B Browse, A List archives, S Search active file and archives,
               SF Search active file, SA Search archives, I File/DSN info
                                                                    Archive
Cmd Rule/Active UNIX file name                    Start Time        Type Avail.
--- ----------------------------------------- ----------------- ---- ------
    *.TRMD*.*.*                                   Empty       N/A   FILE 0
    /var/syslog/2010/09/30/trmd.log
    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    *.OMP*.*.*                                    30 Sep 2010 16:51 FILE 0
    /var/syslog/2010/09/30/omp.log        1
    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    *.OMP*.*.err                                  30 Sep 2010 16:51 FILE 0
```

In the scrollable section (1), the display includes one entry per z/OS UNIX file destination for which the active file is found. Each entry includes rule, active file name, date and time of the first logged message in the active file, archive type, and number of available archives. For each entry, several line commands are available to browse the active file, search at various levels, and so on.

Guide me function

If you do not know which syslogd destination to search, you can use the guide me function (2) to help identify the destination. User ID and job name are available if syslogd is started with the **–u** option. You can enter the facility name and priority or select from lists if you enter a question mark (?).

If you know your component (such as the z/OS load balancing advisor) but do not know to which facility the component writes log messages, you can select the “component to facility name cross reference” function. This function lets you view a list of components and to which facility names they write.

When exiting the guide me function panel, the destination view displays the destinations that match the attributes that you entered in the guide me function marked with arrows (that is, ==>). In Example 1-30, 2 shows where we previously selected OMP* as the job name (1), the guide me function, shown in Example 1-29.

Example 1-29 Selection in the guide me function

```
*------------------------ z/OS CS Syslogd Browser --------------------------*
OPTION ===>

Enter any or all of the following criteria:

  User ID  . . . .==> *         z/OS user ID of logging process
  Job name . . . .==> OMP*    1 z/OS job name of logging process
  Facility . . . .==> UUCP        ? for list
  Priority . . . .==> *           ? for list

For a guide to which facility names the various z/OS components may use,
please select the following option:

  Component to facility name ==>        Select with /

The syslogd browser does not currently support the host specification
criteria that may be used for messages received from a remote syslogd.

Press ENTER to validate input, press END PF key to return to main panel where
matching syslogd rules will be identified with '==>'

The syslogd browser does not currently support the host specification
criteria that may be used for messages received from a remote syslogd.
```

Example 1-30 Destination view: Result of the guide me function

```
Cmd Rule/Active UNIX file name                  Start Time        Type Avail.
--- ------------------------------------------- ----------------- ---- ------
    *.TRMD*.*.*                                 Empty       N/A   FILE 0
    /var/syslog/2010/09/30/trmd.log
    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    *.OMP*.*.*                                  30 Sep 2010 16:51 FILE 0
==> /var/syslog/2010/09/30/omp.log      2
    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 A  *.OMP*.*.err                                30 Sep 2010 16:51 FILE 0
```

Syslogd browser options

If you enter a B for a Cmd entry in the destination view, the active UNIX file displays. Long messages are wrapped into lines that fit the current ISPF screen width. Normal ISPF FIND command support is available and can be used for simple searches in the file that you are browsing.

If you enter an A for an entry at the destination view (shown at **A** in Example 1-30 on page 29), the available archives displays, as shown in Example 1-31. You can delete individual archive data sets from this panel.

Example 1-31 Archive overview

```
*------------------------ z/OS CS Syslogd Browser ----------- Row 1 to 5 of 5
OPTION ===>                                                  Scroll ===> PAGE

Rule . . . . . . . . . . *.OMP*.*.err
Active file name . . . . /var/log/2010/09/30/omp.err
Current config file. . . /etc/syslog.conf

Press ENTER to select an entry, press END PF key to return to main panel

Line commands: B Browse archive, S Search archive, D Delete archive
               I File/DSN info

Cmd Archive                                          Loc    Start time
--- ------------------------------------------------ ------ ------------------
    /var/log/2010/10/30/omp.err                      *FILE* 30 Sep 2010 00:01
   /var/log/2010/10/29/omp.err                       *FILE* 29 Sep 2010 15:12
```

By entering an S for an entry at the destination view, the search interface opens, as shown in Example 1-32. The search options govern the search operation. The resulting data set name can be an existing data set. If it does not exist, it is allocated using the specified UNIT name, which is initialized according to your corresponding ISPF allocation unit. After the search, you can keep the resulting data set, delete it, or have a standard ISPF print dialog displayed.

All search arguments are optional. A time value **1** must be accompanied by the corresponding date. A date can be entered without a time. (The default from time is 00:00:00, and the default to time is 24:00:00.) All specified search arguments are logically ANDed together.

If you enter specific search criteria and a message has no value for those criteria, the message is considered a *non-hit*. For example, if you specify a user ID, but a message has no user ID, that message is a non-hit, which might be the case if syslogd is not started with the **–u** option,

For local messages, host name is the host name that is configured in TCPIP.DATA.

Example 1-32 The syslogd browser search panel

```
Enter your search options.

  Case sensitive  ==> NO         (Yes/No) Are string arguments case sensitive?2
  Maximum hits    ==> 5          (1-99999) Max number of hits to display
  Result DSN name ==> 'CSO2.SYSLOGD.LIST'
  Result DSN UNIT ==> SYSALLDA   Unit name for allocating new result DSN
  Result DSN disp ==> 1          1:Keep, 2:Delete, 3:Display print menu

Enter your search arguments.  All arguments will be logically ANDed.

  From date  . . .==>            (yyyy/mm/dd) Search from date
  - and time . . .==>            (hh:mm:ss) - and time (24-hour clock) 1
  To date  . . . .==>            (yyyy/mm/dd) Search to date
  - and time . . .==>            (hh:mm:ss) - and time (24-hour clock) 1
```

```
  User ID  . . . .==>            z/OS user ID of logging process
  Job name . . . .==>            z/OS jobname of logging process
  Rem. host name .==>
  Rem. IP address ==>
  Message tag  . .==> omproute        Enter ? for list 3
  Process ID . . .==>            z/OS UNIX process ID
  String 1 . . . .==>
  String 2 . . . .==>
  String 3 . . . .==>
  String 4 . . . .==>

Message tags are typically component names.  PID availability depends on
options set by the logging application.  UserID and Jobnames are available
for local messages if syslogd is started with the -u option.
```

Note: In some cases, message text case does matter 2. If you say `NO` to `Case sensitive` and then search for a string of `abc`, messages with “ABC,” “abc,” “Abc,” and so on are considered hits. If you specify `YES` to ‘`Case sensitive` and then search for a string of `abc`, only messages with the exact matching case “abc” are considered hits. Note the case sensitivity option only applies to the four free-form string fields.

If you need to limit your search to a specific component that is not easily isolated by job name, try to find the message tag 3 the component uses. By entering a question mark (?) in the component tag field on the search panel, all known component tags display and you can choose the one in which you are interested, as shown in Example 1-33. Remember that it is optional for an application to include a component tag in logged messages. However, all known z/OS applications that log to syslogd do include a message tag.

Example 1-33 syslogd browser search results

```
z/OS CS Syslogd Browser Search Results - Date: 30 Sep 2009 Time: 16:38:20

Case sensitive  . . . NO
Max. number of hits . 5
Syslogd Config  . . . /etc/syslog.conf
Searched files/DSNs . 6
    File/DSN  . . . . /var/log/2010/09/30/omp.err
    File/DSN  . . . . /var/log/2010/09/29/omp.err

Search Arguments:

    From date . . . .
     and time . . . .
    To date . . . . .
     and time . . . .
    User ID . . . . .
    Job name  . . . .
    Remote host name.
    Remote IP addr. .
    Message tag . . . omproute
    Process ID  . . .
    String 1  . . . .
    String 2  . . . .
    String 3  . . . .
    String 4  . . . .
```

```
Line no. File or data set: /var/log/2010/09/30/omp.err
******** ************************************************************
00000001 Sep 30 16:51:54 WTSC30/TCPIP    OMPB     omprouteÝ131220¨:
           EZZ8107I OMPB subagent: Connection to SNMP agent Dropped

00000002 Sep 30 16:51:54 WTSC30/TCPIP    OMPB     omprouteÝ131220¨:
         EZZ7842I Informational recv() error, errno=122:EDC5122I
         Input/output error., errno2=746d0381

00000003 Sep 30 16:51:54 WTSC30/TCPIP    OMPB     omprouteÝ131220¨:
         EZZ7842I OSPF recvfrom() error, errno=122:EDC5122I
         Input/output error., errno2=746f0381

Line no. File or data set: /var/log/2010/09/29/omp.err.
******** ************************************************************
```

Diagnosing the syslogd browser

Debug information is available. Both EZABROWS and EZASYRGO support a debug keyword with two numeric values ranging from `0` (no debug) to `9` (a lot of debug).

The browser can be started with a debug option as follows:

```
Debug(N1,N2)
```

In this command:

- N1: Sets the debug level for main components
- N2: Sets the debug level for BRIF driver component

The debug levels are from 0 to 9 as follows:

0: No debug (the default if the debug option is not specified)
1: Minimal debug
5: Medium debug amounts
9: Very verbose debug

Note: Debug level 9 produces *many* debug messages.

By default, debug output goes to the terminal. You can redirect debug output to data sets.

Important: Use debug options only if instructed to do so by z/OS CS support personnel.

The use of the syslogd browser is optional. It displays and searches syslogd messages that are generated by z/OS. Most capabilities will work with log messages received from other platforms. Some inconsistencies might exist with the format of the actual messages, which can confuse the browser when searching. Be sure that all syslogd log files referred to in the syslogd configuration exist because data cannot be returned if any log file is missing.

When browsing syslogd files that are collected by a network syslogd from a remote system, keep in mind the following considerations:

- Make sure this system has access to the z/OS UNIX file system of the other system.
- If system symbols are used in the data set prefix option in the syslogd configuration file, consider using the override option on the initial browser panel.

Composition of a message logged by syslogd

To have all messages logged with your local time, set the TZ environment variable in the CEEPRMxx PARMLIB member. When you do so, you need to define the TZ environment variable for all three IBM Language Environment® option sets (CEEDOPT, CEECOPT, and CELQDOPT), as shown in Example 1-34.

Example 1-34 Coding for time zone in CEEPRMxx

```
CEECOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
CEEDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
CELQDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
```

For local messages, host name is the host name that is configured in TCPIP.DATA.

In Example 1-35, for the timestamp 1, the month is always a 3-character English month name followed by the day in the month. Note that syslogd never includes the year. Time of day is always in 24-hour clock format. You can control the time value using the TZ environment variable.

Example 1-35 shows the host 2, the user ID 3, the job name 4, the TAG 5, the PID 6, and text 7. The user ID and job name are available for local messages when syslogd is started with the **–u** flag. The message tag is an optional character string that can be passed by the logging application to identify the application or component that created this log message.

Example 1-35 Message logged by syslogd

```
Sep 30 16:51:54 1 WTSC30 2 /TCPIP 3 OMPB 4 omproute5Ý131220 6¨: EZZ8107I7
OMPB subagent: Connection to SNMP agent Dropped
```

1.5 Problem determination for syslogd logging

If a log file does not contain any messages, first make sure the job is running. If the job is running, restart syslogd (with SIGHUP or with F SYSLOGD,RESTART) and add **-u** as an additional parameter. The **-u** parameter adds the user ID and job name information to the message. Add a new entry in the syslogd configuration file to send all messages to a debug file as shown in Example 1-36.

Example 1-36 Creating a debug entry to save all messages

```
wtsc31.OMP*.*.debug  /var/syslog/%Y/%m/%d/omp.debug -F 640 -D 770
*.*                  /var/syslog/%Y/%m/%d/all.debug -F 640 -D 770
```

After you run syslogd again with the **-u** parameter, check the log file to determine if perhaps the job-specific log file is empty, because the user name does not match the user name specified in `/etc/syslog.conf`. Example 1-37 on page 34 shows a section of our messages file where OMPB was started with the wrong user name.

Example 1-37 Entries in log file with incorrect user name

```
Sep 30 23:39:51 WTSC30/TCPIP 1   OMPB 2   omprouteÝ67240183¨: EZZ7800I OMPB
starting
Sep 30 23:39:51 WTSC30/TCPIP    OMPB     omprouteÝ67240183¨: EZZ7845I Established
affinity with TCPIPB
Sep 30 23:39:51 WTSC30/TCPIP    OMPB     omprouteÝ67240183¨: EZZ7817I Using
default OSPF protocol 89
```

The field immediately following the date appears as a result of the **-u** parameter and identifies the system and user ID **1** that started the job. The next field also appears as a result of the **-u** parameter and identifies the job name **2**. The next field after the job name is the beginning of the message. As shown in Example 1-37, `OMPB` was started using the user name `WTSC30/TCPIP` **1** and the syslogd configuration incorrectly used only the system identification as the user name, which explains why omp.debug file is empty.

For other problems, start syslogd with the **-d** parameter to enable a debug trace. Output from the debug trace is sent to the stdout stream that is defined in your MVS JCL or in the UNIX command line executable. (See examples of stdout and stderr output in Example 1-5 on page 11, Example 1-6 on page 11, and Example 1-7 on page 12.)

Warning: If you enable debugging, remember from the discussion in “Use a syslogd environment variable file” on page 12 that the default debugging level of 127 produces copious amounts of output. You might want to consider coding a lower value for the SYSLOGD_DEBUG_LEVEL environment variable. (You can specify this variable in the syslogd environment variable file or with an export statement for syslogd initialization specified in `/etc/rc`.)

1.6 Additional information sources for syslogd

See the following sources for additional information:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- RFC 3164

Tip: For descriptions of security considerations that affect specific servers or components, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

2

TN3270E Telnet server

Telnet 3270 (TN3270) is an SNA 3270 terminal emulation technology that supports access to SNA applications on mainframe computers using TCP/IP (Telnet) protocols. This chapter focuses on the TN3270 functions that are available on the z/OS Communications Server.

A z/OS Communications Server TN3270E Telnet server can be implemented in your mainframe environment to allow TN3270 clients to access SNA applications using TCP/IP (Telnet) protocols. A TN3270 client can be installed and configured independently on each desktop, or a Java Applet version known as Host On-Demand can be used to automate the process. This chapter discusses the following topics.

Section	Topic
2.1, “Conceptual overview of the TN3270E server” on page 36	The basic concepts of TN3270E servers.
2.2, “TN3270E server in a single image” on page 43	A TN3270E basic implementation scenario in a single-image environment, including description, configuration, activation, and verification.
2.3, “Multiple TN3270E servers in a multiple image environment” on page 77	A multiple TN3270E servers implementation scenario in a sysplex environment, including description, configuration, activation, and verification.
2.4, “Multiple TN3270E servers using LU name server and LU name requester” on page 95	A multiple TN3270E servers using LUNS and LUNR implementation scenario in a sysplex environment, including description, configuration, activation, and verification.
2.5, “TN3270E server in a single image using SHAREACB” on page 128	This changes the basic scenario to use SHAREACB configuration statement to reduce CSA utilization
2.6, “TN3270 support of TSO logon reconnect” on page 132	Description of TN3270 support of TSO logon reconnect.
2.7, “Problem determination for the TN3270E servers” on page 132	Problem determination methods for TN3270E server environments.
2.8, “Additional information sources for the TN3270E server” on page 141	References for TN3270E implementations.

2.1 Conceptual overview of the TN3270E server

As illustrated in Figure 2-1, TN3270E Telnet server is one of the standard applications provided with the z/OS Communications Server. It uses z/OS UNIX socket application programming interfaces (APIs), a logical file system (LFS) and a physical file system (PFS) to transfer data from application layers to transport layers, and to manage incoming client requests.

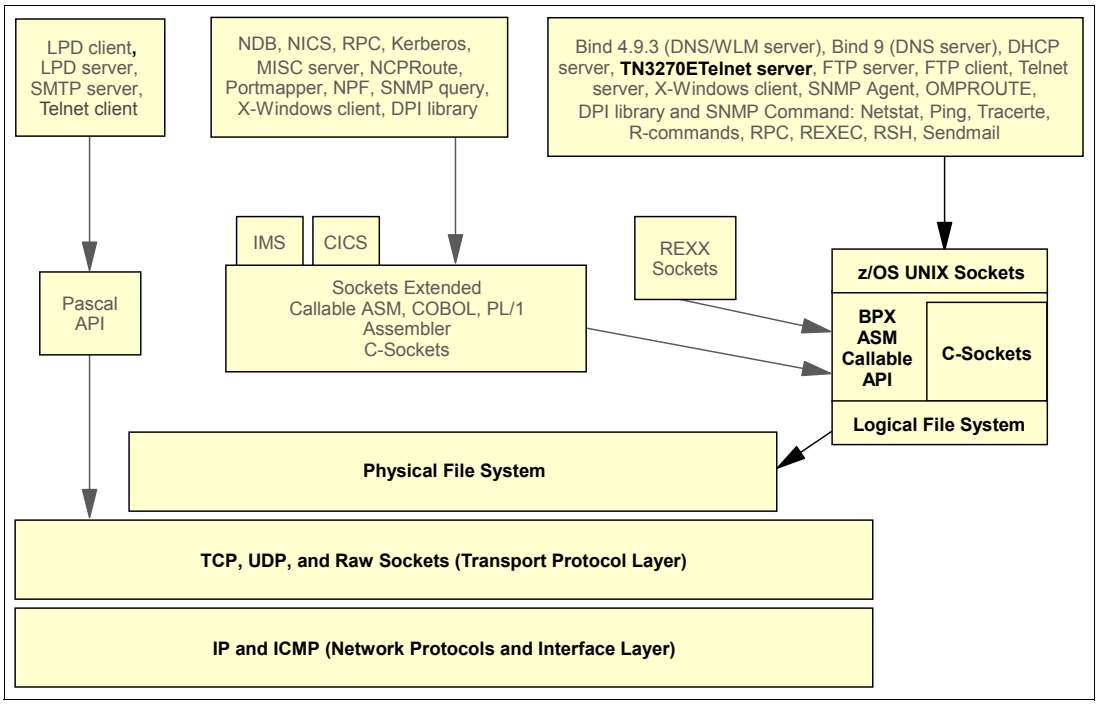


Figure 2-1 z/OS TN3270E Telnet server application services

Terminology: In this chapter, we refer to the TN3270E Telnet server as the *TN3270E server*.

This section discusses the following topics:

- What is the TN3270E server
- How does the TN3270E server work
- Possible uses for the TN3270E server

2.1.1 What is the TN3270E server

Telnet is an IP terminal emulation protocol that enables you to log on to a remote system as though you were directly connected to it. Telnet enables users to have access to applications running on that system. The TN3270E Telnet server provides access to IBM VTAM® SNA applications on the z/OS host using 3270 or linemode protocol. Traditionally, non-mainframe ASCII-based platforms have used the Telnet emulation protocol for achieving this connectivity. However, because the mainframe is an EBCDIC-based platform with special 3270 terminal-type data stream requirements for its connected terminals, the TN3270 function was developed to support mainframe data streams. The TN3270E Telnet server has implemented these special terminal data stream requirements for clients using the 3270 emulation protocol.

Traditional Telnet and TN3270E differ in two main areas:

- 3270 terminal emulation uses block mode rather than line mode.
- 3270 terminal emulation uses the EBCDIC character set rather than the ASCII set.

The z/OS server is an EBCDIC character set based platform. Most other non-z/OS platforms are ASCII character set based. Normal data transmission between these dissimilar platforms requires character translation processes for each packet of data they exchange. TN3270 provides end-to-end 3270 data stream emulation capability. The client performs any necessary conversion between ASCII and EBCDIC character sets. There is a set of enhancements to the base TN3270 support called TN3270E, which is now widely used.

Note: The meaning of the *E* as used in TN3270E and in IBM terminal devices such as an IBM-3279-2-E can be confusing. In both cases, the *E* represents extended capabilities. However, there is no correlation between the extended capabilities of TN3270E and those of an IBM-3279-2-E display station.

327x device types that end in *E* are terminals that support extended field attributes, such as color and highlighting. TN3270E server support includes a number of important enhanced capabilities over TN3270, which we discuss in greater detail in “TN3270E functionality” on page 38.

In addition, z/OS TN3270E server supports RFC 1647 TN3270 Enhancements (July 1994) and RFC 2355 TN3270 Enhancements (June 1998), but not the RFC 1646 TN3270 Extension for LUname and Printer Selection (July 1994). If you want to migrate from a channel-attached router or Communications Server for IBM AIX®, Linux, and Windows to z/OS TN3270E server, the older version of the TN3270 emulator software might need to be replaced.

For descriptions of security considerations that affect specific servers or components, see the *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

2.1.2 How does the TN3270E server work

The TN3270E protocol provides an efficient means for transporting SNA 3270 traffic over an IP network while enabling the migration of user access to TCP/IP-only connectivity, and away from native SNA networking support. The TN3270E server acts as a VTAM application that activates one VTAM application minor node logical unit (LU) to represent each active TN3270E client connection.

Each allocated LU is represented in VTAM by an APPL minor node, which has an ACB control block, created in VTAM’s Private Storage. During the session establishment process, an **`OPEN ACB`** command is issued by Telnet for every LU and, for each **`OPEN ACB`** issued by the TELNET process, VTAM allocates control blocks in ECSA. Environments with a large amount of Telnet clients consume significant ECSA and Private storage.

z/OS Communications Server has implemented the possibility of using a shared ACB for Telnet logical units (LUs) as a way to reduce extended common storage area (ECSA) usage. To get further information about using the SHAREACB option, see 2.5, “TN3270E server in a single image using SHAREACB” on page 128.

Using TN3270E is a two-step process. First the client connects through TCP/IP to the TN3270E server that is listening on some TCP port (usually port 23). Then the TN3270E server allocates VTAM resources and establishes a logical session on behalf of the connecting client.

For quite some time, network gateway devices external to the mainframe have been used to provide TN3270/TN3270E server support for network clients. These outboard servers accessed the mainframe either through the SNA networking infrastructure (such as IBM 3745 Communication Controllers) or through channel connectivity (for example, a channel-attached router). They transformed the client IP connections into SNA sessions for access to the mainframe. Such outboard servers are often limited in functionality, represent an extra point of failure, and keep SNA requirements in the network, which conflicts with efforts to reduce SNA networking requirements. By implementing the TN3270E server on the mainframe, clients can access it using end-to-end native IP without requiring any SNA network support for their terminal traffic.

We discuss important functions that the z/OS TN3270E server supports in the following sections:

- TN3270E functionality
- Connection security
- Connection mapping
- Connection and session management
- Multilevel security compliance
- Connection diagnostics

For complete details, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

TN3270E functionality

Traditional Telnet (line mode, based on ASCII) is limited in functionality when connected to z/OS mainframe SNA applications. Benefits can be achieved by implementing TN3270, and even more by implementing TN3270E. We suggest that you use TN3270E for all mainframe SNA application connections, unless the client simply does not support it.

Figure 2-2 illustrates some of the features of TN3270E.

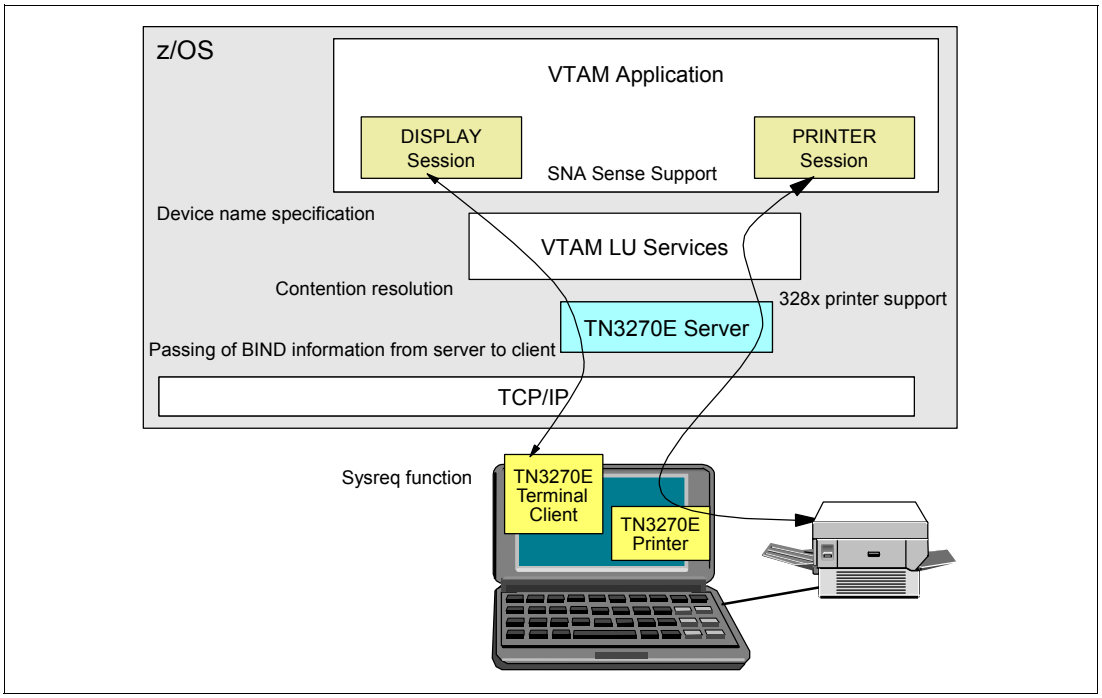


Figure 2-2 TN3270E negotiable features

When the client and the server have finally agreed on using TN3270E through negotiation during the connection process, they negotiate the subset of TN3270E options. These options include device-type and the following set of supported 3270 functions:

- 328x printer support
- Device name specification
- The passing of BIND information from server to client
- Sysreq function
- Contention resolution
- SNA Sense Support

Connection security

Important connection security functions are as follows:

- Data overrun security
  - Protects against a client hung in a send-data loop
  - Protects against using large amounts of storage
  - Limits the number of session requests received by a TN3270E server in a 10-second period
  - Limits the number of chained RUs received without a corresponding end chain RU
  - Avoids auto-reconnect loops upon client connect errors by keeping connection active
- Network Access Control (NAC)
  - Limits user access to certain IP security zones defined by the NETACCESS statement.
  - Manages NAC user ID send and receive access for these security zones
  - NAC user ID is based on the application's address space information
- Transport layer security
  - Secures TN3270E connections with the transport layer security (TLS) or secure sockets layer (SSL) protocol
  - Enables installations to support both types of secure clients without knowing which protocol the client is using
  - SSLV2, SSLV3, and TLS supported (NOSSLV2 is the default.)
  - Enables using Application Transparent Transport Layer Security (AT-TLS) to manage secure connections for TN3270E.

In addition to native TLS support, the TN3270E server can use Application Transparent Transport Layer Security (AT-TLS) to manage secure connections. TLS managed by AT-TLS (TTLSPORT) supports more security functions than TLS managed by the TN3270 server itself (SECUREPORT).

Note: If you have security parameters in a PARMSGROUP statement that are mapped to host names, you cannot emulate that mapping with AT-TLS.

If you are not using PARMSGROUP to map to host names, we urge you to migrate to the use of AT-TLS as a consistent solution for all of your TCP applications.

Be aware of these AT-TLS capabilities and requirements when planning AT-TLS support for TN3270E:

- Dynamically refresh a key ring.
- Support new or multiple key rings.

- Specify the label of the certificate to be used for authentication instead of using the default.
- Support SSL session key refresh.
- Support SSL session reuse.
- Support SSL sysplex session ID caching.
- Trace decrypted SSL data for Telnet in a data trace.
- Receive more granular error messages in syslogd for easier debugging.
- Policy Agent must be active.
- TLS security defined within the TN3270E profile will continue to be available and can be implemented concurrently with AT-TLS.

See *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999 for complete details about implementing TLS and AT-TLS security for the TN3270E server.

Note: The TN3270E client, activated through the TSO TELNET command, does not support any SSL or TLS security protocols. Basic mode is the only method supported by the TN3270E client.

TLS/AT-TLS recommendations

If possible, use AT-TLS as a consistent security solution for all of your TCP-based applications. Certain applications (such as FTP and Telnet), however, have already been programmed to use the SSL/TLS toolkit directly and provide additional security functions such as application-negotiated SSL/TLS and certificate-based user ID mapping.

If, however, you do not need those additional functions, consider implementing FTP and TN3270E with the full AT-TLS support. AT-TLS is the preferred method of implementing TLS support to achieve a consistent solution for *all* of your TCP applications.

Connection mapping

When a client connection request is made, TN3270E must assign an LU name to represent the client. Additionally, an unformatted system services table, an interpret table, a default application, unique parameters, and network monitoring actions can optionally be assigned to the connection. There are 11 mapping statements available in the BEGINVTAM block that map objects to specified client identifiers within the port indicated on the BEGINVTAM block. This robust mapping function gives you flexibility in supporting your organization's requirements for running TN3270E services on the z/OS platform.

Connection and session management

In addition to the basic function of facilitating session setup, TN3270E supports several advanced functions such as the following functions:

- Telnet Solicitor or USS logon panel: provides both Telnet Solicitor panel and unformatted system service (USS) screen support. A Telnet Solicitor panel or USS screen prompts the user to enter the name of the application to log onto. Telnet Solicitor panel is used when no matching USS table is provided and no default application name for the user. The z/OS Communications Server V1R13 adds support for password phrases on the Telnet Solicitor panel for better security control.
- Session initiation management: controls which default application the user should connect to when the session is initialized. It also provides control of whether the user should go

back to the default application login screen or the Telnet Solicitor (USSMSG10) screen when the user has logged off the application.

- Connection and session takeover: assists with lost, disconnected sessions.
- TSO logon reconnect: enables to reconnect even when an old SNA session exists. See 2.6, “TN3270 support of TSO logon reconnect” on page 132.
- Queuing sessions: assists with session manager applications.
- Disconnect on session error: determines type of session termination.
- Bypass RESTRICTAPPL with CERTAUTH: client certificate is used to derive a user ID.
- Allow printer sessions with RESTRICTAPPL: enables printers to establish a session with an application defined as a RESTRICTAPPL, because printers cannot submit a user ID and password.
- The option of keeping the ACB open for an application that needs to INQUIRE for status.
- Express Logon Feature (ELF): TN3270E uses the client certificate to resolve the user ID and IBM RACF® product generates a temporary password, a passticket. ELF requires a secure connection with level 2 client authentication, a client that supports ELF, and RACF passticket setup.
- Cleanup on idle connections: Use the PROFILEINACTIVE parameter statement with a specified time-out value to drop connections associated with a non-current profile that do not have an SNA session.

Connection information passed to VTAM

During session establishment, TN3270 server creates a secondary logical unit (SLU) or Telnet LU name. The TN3270 server creates a control vector CV64 containing additional information such as client IP and Port number. When establishing a session, a VTAM session initiation record CINIT is created and sent to the primary logical unit (PLU). Control vector 64 is appended to this CINIT. In this way, TN3270 server passes connection information to the PLU. If there is a session manager between Telnet and the PLU, the PLU does not receive a CV64, because the session manager receives the CINIT with the CV64, but it cannot send the CV64. The session manager has no SNA APIs available to propagate the CV64 information. Therefore connection information is not delivered to the PLU.[1]

To solve the CV64 forwarding problem, z/OS Communications Server added API support to enable the session manager application to pass the CV64 information to the final PLU. See Figure 2-3, on the SNA session 2, where the session manager sends the CV64 information to the target application.

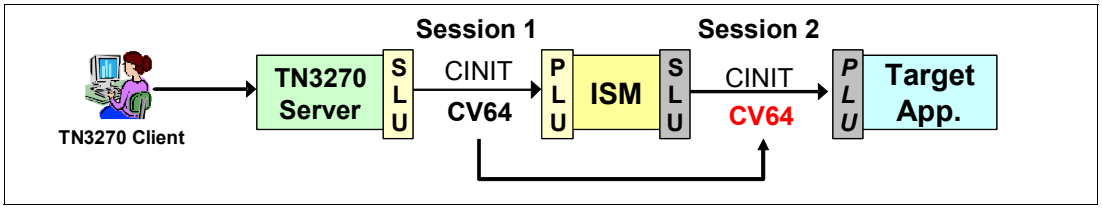


Figure 2-3 ISM sends the CV64 information

Note: The session manager has an SNA API available to propagate the CV64 information. IBM Session Manager (ISM) 2.2.05 and later support this new function.

[1] Exceptions exist. Some session managers “close dest” to set up an end-to-end session between the TN3270 SLU and the application PLU. In this case, the CV64 is passed to the PLU at the final destination. Other types of session managers maintain two sessions: one in which they act as the PLU to the TN3270 SLU, and one where they are the SLU to the application PLU. In this case, the CV64 is not passed to the VTAM with the application PLU.

Multilevel security compliance

The stand-alone TN3270E server can now participate in a multilevel secure environment that uses security labels from z/OS Communications Server. To ensure correct security label comparisons, Network Access Control (NAC) must also be active for Telnet.

For more information about preparing for TCP/IP networking in a multilevel secure environment and NAC, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Connection diagnostics

In addition to general diagnostic tools such as CTRACE and dumps, which are described in *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782 and *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996, several TN3270E-specific diagnostic tools are available:

- Debug messages: TN3270E-specific debug messages can be turned on or off to diagnose TN3270E problems. Several types of debug messages are available.
  - Summary messages indicate important status changes.
  - Detail messages indicate that a problem was detected.
  - General messages indicate an important event.
  - Trace messages show data to and from the client, and to and from VTAM.
  - Flow messages show entry into modules and exit from modules.
  - MSG07 enables TN3270E to send a message to the client indicating an error occurred and what the error was.
- The ABENDTRAP command can be used to set up an abend based on the variables specified.
- Optional SMF recording is controlled by using the SMFINIT and SMFTERM statements.
- TESTMODE causes the profile syntax to be checked without making the profile active.
- Several displays are available that provide summary and detail information.
- TCP/IP Packet Trace using the system's CTRACE facility (SYSTCPDA).
- TCP/IP Socket Data Trace using the system's CTRACE facility (SYSTCPDA).
- TCP/IP Component Trace using the system's CTRACE facility (SYSTCPIP).

See 2.7, "Problem determination for the TN3270E servers" on page 132 for more detailed examples.

2.1.3 Possible uses for the TN3270E server

You can use the TN3270E server in the following scenarios:

- TN3270E server in a single image
- Multiple TN3270E servers in a multiple image environment
- Multiple TN3270E servers using LU name server and LU name requester

These three scenarios are commonly implemented on single-image and sysplex environments. For other advanced TN3270E high availability or security implementation scenarios, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998 and *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

Note: The TN3270E server running within the TCP/IP stack is no longer supported.

If you are migrating from a previous release that supported running the TN3270 server within the stack's address space, you must migrate the TN3270E server to its own address space.

Migration tip: Here is an easy way to prepare the TN3270E profile configuration to facilitate moving it from running within the TCP/IP stack to running external to the stack, using the following steps:

1. Place all the TN3270-related statements into a separate configuration member. (For example, you could call it PROFTELN.) Remove all TN3270-related statements from the TCP/IP stack configuration member (such as PROFSTAK).
2. Point the TN3270 task's //PROFILE DD to the TN3270 profile member, PROFTELN. (It contains TN3270-related statements only.)
3. Remove the INCLUDE statement for the PROFTELN member in your PROFSTAK, if you used the statement previously. Insert a TCPIPJOBNAME statement into the TELNETGLOBALS block to achieve the stack affinity with the TCP/IP stack that this Telnet server ran with before. See the explanation of TCPIPJOBNAME in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.
4. Change INTCLIEN to the TN3270E server's procedure name on the port reservation statements along with the NOAUTOLOG keyword. Remove NACUSERID (which was required for Network Access Control of Telnet connections when Telnet ran in the TCPIP address space). NACUSERID is optional, not required.
5. Use the TESTMODE parameter when you validate the new TN3270 profile to report the latent errors, then correct them. When no errors are reported, remove the TESTMODE parameter.

See Appendix B, "Sample files provided with TCP/IP" on page 407, for sample files available for use with the TN3270E server.

2.2 TN3270E server in a single image

This section provides an overview of executing the TN3270E server in one MVS image and includes the following topics:

- Description of our TN3270E server scenario
- Configuration of the TN3270E server
- Activation of the TN3270E server

2.2.1 Description of our TN3270E server scenario

In this scenario, we use one system image, one TCP/IP stack, and one TN3270E server running in its own address space. Stack affinity is established for the TN3270E server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block. The relationship between the TN3270E server, the TN3270 client, and the TCP/IP stack is illustrated in Figure 2-4.

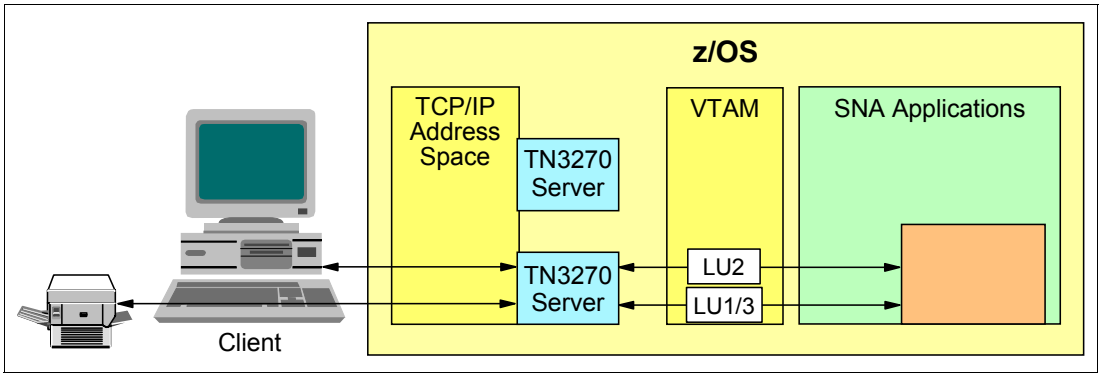


Figure 2-4 TN3270E Telnet server executing in an MVS image

As shown in Figure 2-4, you can run one or multiple TN3270E servers in a single-image system, each in its own address space. In this section, we discuss the implementation of one TN3270E started task. In the same way, you can set up multiple separated TN3270E servers in one LPAR.

Our TCP/IP stack started task name is *TCPIPB*. The TN3270E started task name is *TN3270B* on system SC31.

2.2.2 Configuration of the TN3270E server

Perform the following tasks to configure the TN3270E server:

- Customize the VTAM APPL major node for the TCP/IP LU names.
- Add the VTAM APPL major node to the VTAM startup configuration.
- Customize the TCPDATA configuration data set.
- Design the mapping of APPL and LU assignments.
- Understand connection mapping terminology.
- Adopt a connection mapping methodology.
- Customize the TN3270 PROFILE data set.
- Define system security for the TN3270 started task.
- Consider program properties table attributes for the TN3270E server.
- Customize the JCL for the TN3270 started task.

Customize the VTAM APPL major node for the TCP/IP LU names

You can find a sample VTAM APPL major node in SEZAINST(IVPLU). See *z/OS Communications Server: IP Configuration Guide*, SC31-8775 for an in-depth discussion about how to prepare the VTAM LU definitions for the TN3270E server. Give attention to the values that are coded for the following keywords on the VTAM APPL statement:

- `SESSLIM=`
- `EAS=`
- `PARSESS=`

Some SNA applications do not issue CLSDST when their LOSTERM exit is driven, which can create a hang condition for the TN3270E LU that has issued a CLOSEACB and is waiting for an UNBIND RESPONSE from the application.

Note: Code `LOSTERM=IMMED` on all target (PLU) applications that will have an SNA session with TN3270E to avoid CLOSEACB hang conditions. Code `EAS=1` to minimize Common Service Area (CSA) storage use. Code `PARSESS=NO` for parallel sessions should not be used with Telnet LUs. Code `SESSLIM=YES` because Telnet server LUs do not support multiple concurrent sessions.

These LU names represent each TN3270E client IP connection. It is this VTAM LU that logs on to a selected VTAM application. The TN3270E server uses application LUs that are defined in VTAM application (APPL) major nodes to represent clients, by making them look and act the same as VTAM terminal LUs. These APPL definition members must be made available to VTAM by being in one of the data set that is specified with the VTAMLST DD statement in the VTAM started JCL. Add the APPL definition members in to the ATCCONxx member of VTAMLST to ensure that the TN3270E Telnet server applications are activated when VTAM is started,

You can enter an APPL statement for each LU, or you can use a model APPL statement. Use a model statement to avoid all the clerical effort of maintaining a large list and to minimize storage utilization. (The storage for that APPL and its ACB are allocated only when the LU is in use.) There is an extensive discussion on coding techniques for the model APPL statement in *z/OS Communications Server: SNA Resource Definition*, SC31-8778. Our model statement uses an asterisk (*) in the name as a wild card. You can also use system symbolics to assist when multiple systems are involved and you want them to share the VTAMLST and the same member within that VTAMLST. Example 2-1 shows one technique for using system symbolics.

Example 2-1 Sample APPL model major node used in our scenario

```
BROWSE    SYS1.VTAMLST(@TCPLUS) - 01.01             Line 00000000 Col 001 080
TCPLUS&SYSCLONE. VBUILD TYPE=APPL
&SYSNAME.B* APPL  AUTH=NVPACE,                                          X
               EAS=1,                                                   X
               PARSESS=NO,                                              X
               SESSLIM=YES,                                             X
               MODETAB=ISTINCLM
```

The &SYSCLONE value (30, 31, and 32 in our test environment) is used to generate the label on the VBUILD statement, and the &SYSNAME value (SC30, SC31, SC32 in our test environment) is used to generate the actual APPL model name. When the APPL model major node is activated by a particular VTAM, the fully generated unique names would look similar to those shown in Example 2-2. The actual name assigned for a connection is determined by the TN3270E server mapping rules, and will be assigned at the time of connection negotiation.

Example 2-2 VTAM APPL names resulting from activating major node using system symbolics

```
SC30BB01 thru SC30BB99 when TN3270 DEFAULTLUS specify SC30BB01..SC30BB99
SC30BS01 thru SC30BS99 when TN3270 DEFAULTLUS specify SC30BS01..SC30BS99
and
SC31BB01 thru SC31BB99 when TN3270 DEFAULTLUS specify SC31BB01..SC31BB99
SC31BS01 thru SC31BS99 when TN3270 DEFAULTLUS specify SC31BS01..SC31BS99
```

Add the VTAM APPL major node to the VTAM startup configuration

To activate the application definition major node automatically, include it in ATCCONxx. If multiple TN3270E servers are used (for example, multiple TCP/IP stacks in a sysplex environment), and if you do not use LU name server (*LUNS*) to coordinate LU name in a sysplex, ensure that each server uses unique LU names. Otherwise, the second server that uses the same LU name will not be able to establish a session. Either the OPEN ACB request will fail or the cross-domain session request will fail.

Note: Use the LUNS to centralize LU name allocation and avoid duplicate LU name assignments among the group of TN3270E servers known as LU name requesters (or *LUNR*). See 2.4, “Multiple TN3270E servers using LU name server and LU name requester” on page 95 for a detailed description about LU name server and LU name requester.

Customize the TCPDATA configuration data set

The TCPDATA data set is normally customized during basic stack setup and initialization. For details about customizing the `TCPDATA` file and the resolver, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996. TN3270 uses the native MVS and not the UNIX sockets search order to find a resolver. If you have deployed a Global Resolver file, you simplify the search for a resolver file, because the distinctions between the native MVS and UNIX sockets search order become irrelevant. If no resolver is defined, no host name is found.

Design the mapping of APPL and LU assignments

When a client connection request is made, TN3270 must assign an LU name to represent the client. Additionally, an unformatted system services (USS) table, an interpret table, a default application, unique parameters, and network monitoring actions can optionally be assigned to the connection. There are 11 mapping statements available in the BEGINVTAM block that map objects to specified client identifiers within the port indicated on the BEGINVTAM block:

- Five are application setup related.
- Four are LU name assignment related.
- One maps connection parameters.
- One maps monitoring rules.

Shortly after a connection request is accepted, the mapping statements are used by TN3270 to map, or assign, as many of the objects to the client as possible. The set of assigned objects is used for the duration of the connection. These mappings are accomplished by performing the following processes:

- Mapping objects to client identifiers
- Application name mapping
- USS and interpret table mapping
- LU name mapping: Generic or specific
- Printer name mapping: Generic or specific
- Connection parameters mapping
- Connection monitoring mapping

The NULL set of clients is defined as that group of clients for which no explicit mapping statement applies. Mapping statements that do not require an assignment to a specific client ID can be used to set the default assignments for the NULL client group, by simply omitting the client ID from the statement. The following mapping statements can omit the client ID:

- DEFAULTAPPL
- PRTDEFAULTAPPL
- LINEMODEAPPL
- USSTCP
- INTERPTCP

The following additional statements names imply a mapping association to the NULL client group:

- DEFAULTLUS/SDEFAULTLUS
- DEFAULTLUSSPEC/SDEFAULTLUSSPEC
- DEFAULTPRT/SDEFAULTPRT
- DEFAULTPRTSPEC/SDEFAULTPRTSPEC

These uniquely named groups do not require mapping, because their names imply it.

For complete details about mapping objects to client identifiers, see that topic in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. A complete statement syntax is shown in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. These two books provide comprehensive discussions and examples related to mapping. We do not intend to duplicate the discussions or examples in this book. However, the following discussion might help you comprehend the text in the books that are referenced.

There are 15 object types that can be mapped to 10 client ID types. There are 9 object types that can be mapped to the NULL client ID. Therefore, the total number of unique mapping combinations supported is 159. It is not practical to attempt to show an example of each one of these mappings. However, we do want to discuss a methodology you can use to define a valid mapping strategy. Obviously, the scenario that yields the least amount of clerical effort in customizing a TN3270 configuration file is the one that does not require any explicit mapping assignments. In many cases, a TN3270E server can assign all connecting clients to an LU name from the DEFALUTLUS pool, and force them all to the same DEFAULTAPPL or to the same USSTCP table. If this meets your organization's needs, then customizing the TN3270 profile data set will be minimal effort for you.

If you do need to establish some level of mapping assignments, we strongly suggest that you do not define unnecessary mapping just because the capability is there. Do not get carried away. Too much of a good thing can turn out to be a clerical effort that you do not want to maintain in the future. Establish only those mappings that are required, and keep it simple. With this thought in mind, we use only a few mapping statements, objects and client IDs in our examples to illustrate the mapping capabilities.

Understand connection mapping terminology

Individual objects and individual clients do not need to be defined prior to referring to them in a mapping statement, as shown in Example 2-3.

Example 2-3 Mapping an individual object or an individual client ID

```
DEFAULTAPPL CICSCLP0 10.1.8.21
LUMAP SC30BB74 USERID,CS07
```

However, groups must be defined prior to referring to them in a mapping statement, which applies to both object groups and client ID groups. We define a few terms in Table 2-1.

Table 2-1 Group types used in mapping statements

Type of group	Group description
Object group	A list of objects of the same object type having something in common
Parameter group	A list of options overriding the defaults set in TELNETPARMS
Monitor group	A list of options defining monitoring actions
Client group	A list of Client IDs of the same client type having something in common

The format of a mapping statement is shown in Example 2-4.

Example 2-4 Mapping statement format

```
Mapping Keyword   Object specification   ClientID specification
```

Adopt a connection mapping methodology

Groups must be defined before they are referred to in a mapping statement, and on the mapping statement, the object is specified followed by the client ID. With these things in mind and for consistency, organize your group definitions in the TN3270 profile configuration data set in the following order:

1. Object groups
2. Parms groups
3. Monitor groups
4. Client ID groups
5. Mapping statements

This is not a requirement, but it does assist in the management and readability of the definitions. By arranging the definitions in a consistent order you help document the environment. Others who later have to maintain the configuration will be able to more easily understand your design.

Some objects must be specified as the only object being mapped to a client ID or client ID group. They cannot be specified in a group. These object types are VTAM APPLs, USS tables, and interpret tables, and are specified on the DEFAULTAPPL, PRTDEFAULTAPPL, LINEMODEAPPL, USSTCP, and INTERPTCP statements. However, the remaining object types can be specified in a group. This includes defining them as the only member of a group.

Any single client ID can be specified as the only member of a client ID group.

Tip: To establish a consistent grouping methodology, consider using the technique we describe here. Where the rules allow, use all *group* definitions, even for single objects and client IDs that you need to map. The mapping statements will then always be referring to group names (where the syntax permits), and not to single objects or clients. Even if the group consists of only a single item, your mapping statements will be consistent.

This technique offers an additional advantage. That is, if you ever have to add an object or a client ID to the *existing* single item being mapped, its group is already defined, and you avoid the inconvenience of creating a group and changing the mapping statement to point to it. The group is already set up—you can just add to it.

Customize the TN3270 PROFILE data set

A sample configuration can be found in SEZAINST(TNPROF).

The PROFILE data set (usually a PDS member) contains all the necessary statements to define the TN3270 environment to the server. The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, discusses how to use the statements to accomplish the support your environment requires. The statements, their parameters, and statement syntax are discussed in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

For this scenario we created a new profile in TCPIPB.TCPPARMS, named TELNB31A.

The purpose of the TN3270 configuration statements is to:

- Define connection characteristics.
- Facilitate session setup with host VTAM applications.
- Assign an LU name to represent the client connection.

The TN3270 configuration uses the following statement blocks:

- TELNETGLOBALS/ENDTELNETGLOBALS
  - An *optional* statement block containing TN3270 parameter statements.
  - The parameters define connection characteristics for *all* ports.
- TELNETPARMS/ENDTELNETPARMS
  - A *required* statement block containing TN3270 parameter statements.
  - The parameters define connection characteristics for a *specified* port.
- BEGINVTAM/ENDVTAM
  - A *required* statement block containing TN3270 mapping statements.
  - Mapping statements define how applications and LU names are assigned to clients.
- PARMSGROUP/ENDPARMSGROUP
  - An *optional* parameter group statement within the BEGINVTAM group containing group characteristics statements. The parameters define connection characteristics for the mapped clients.

Stack affinity issues

When TN3270 runs as its own procedure, stack association depends on whether the TCPIPJOBNAME statement in the TELNETGLOBALS block is used. Even in an INET environment where only a single TCP/IP stack can run, TCPIPJOBNAME must be specified for TN3270 to be associated with that stack for some functions.

TN3270 connections automatically associate with the active stack, but the functions listed here do not work if TCPIPJOBNAME is not explicitly specified:

- TN3270 SNMP subagent activation requires specification of a stack name to register with the agent. Without TCPIPJOBNAME, TN3270 blocks the subagent activation request.
- WLM registration requires specification of a stack name for successful registration. Without TCPIPJOBNAME, a profile DEBUG message is issued if WLM registration is attempted.
- Netstat displays might not show all TN3270 connections if multiple stacks are supporting the server. Only those connections supported by the stack where the command is issued are shown.

In a common INET (CINET) environment, TN3270 is, by default, associated with all running stacks. If another stack is started while TN3270 is active, the current LISTEN for the port is

cancelled and reissued automatically to include the new stack. If association with one stack is desired for control purposes or for functionality support, specify TCPIPJOBNAME.

TN3270 SNMP subagent limitation

The TN3270 SNMP subagent can only register with one agent, and each agent can support only one TN3270 subagent. If running TN3270 in its own address space, in addition to the required affinity, you must be careful to plan for one agent per TN3270 subagent.

Note: If multiple TN3270 SNMP subagents initialize to the same agent, the agent forwards all data requests to the first subagent that connected, and all other initializations are queued. If the first subagent ends, the next subagent in the queue then receives all data requests. This is probably not the way you would have expected or wanted it to work.

SMF address space name field

When running TN3270 as its own procedure, the address space name or started task name is the name of the TN3270 procedure.

Note: When you migrate your TN3270E server from running within your stack task to its own address space, then you need to update your SAS, MICS, or other data reduction programs that might be using the started task procedure name to identify records.

With the TN3270E server within the stack, the programs are accustomed to seeing the stack task name. Now, with the change to its own address space, there will be a new name in the field, and the code might have to be changed.

Port reservation

Reserve a TN3270 port in TCP/IP stack with its procedure name, because INTCLIEN is not supported any more.

Note: If the TN3270 port is reserved, it must be reserved by specifying the stand-alone TN3270 procedure name on the PORT reservation statement:

```
PORT 23 TCP TN3270B NOAUTOLOG
```

A portion of the TN3270 configuration profile data set is shown in Example 2-5. See Appendix C, “Configuration files: TN3270E stand-alone scenario” on page 415 to review the complete profile.

Example 2-5 TN3270B configuration profile (TELNB31A) for stand-alone task

```
BROWSE    TCPIPB.TCPPARMS(TELNB31A) - 01.02            Line 00000000 Col 001 080
; ===SC31============ TN3270 Server Profile for Standalone Task =======-
;                                                                       -
;   No SSL security.  No Sysplex Distribution in the stack.             -
;                                                                       -
; ---------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
    . . . . . . . . . .
ENDTELNETGLOBALS
;
TELNETPARMS
    PORT 23
```

```
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC31BB01..SC31BB99
  ENDDEFAULTLUS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL SC*             ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION,5 ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
```

For complete detailed listings of the started task procedures and profiles that we used for this scenario, see Appendix C, “Configuration files: TN3270E stand-alone scenario” on page 415.

Define system security for the TN3270 started task

Before TN3270 can be started, security for the procedure name and its associated user ID must be defined. See the TN3270 chapter in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for information about setting up security for the started task. Also review the sample file SEZAINST(EZARACF), which contains sample security statements for this effort.

Note: This discussion assumes RACF is the security subsystem being used. If another security product is used, see its publications for equivalent setup instructions.

The procedure name must be added to the RACF STARTED class and have a user ID associated with it as follows:

```
RDEFINE STARTED TN3270*.* STDATA(USER(TCPIP))
SETROPTS RACLIST(STARTED) REFRESH
```

Coding the started task name using the wildcard format enables us to run multiple TN3270 started tasks without having to define each one separately. Their names are all spelled TN3270*x*, where *x* is the qualifier. They can all be assigned to the same user ID.

Use an existing superuser ID, or define a superuser ID to associate with the job name by adding a user ID to RACF and altering it to superuser status as follows:

```
ADDUSER TCPIP
ALTUSER TCPIP OMVS(UID(0) PROGRAM ('/bin/sh') HOME('/'))
```

In this example the user ID name is *TCPIP*, but any name can be used. You can combine these two RACF commands into one command by putting the OMVS parameter on the ADDUSER command line. The add and alter commands are performed separately in case the user ID already exists. If the add fails, the alter still succeeds.

If setting up a superuser ID is not desirable, you can instead permit the user ID to the BPX.SUPERUSER class using the following steps:

1. Add the user to RACF:

   ```
   ADDUSER TCPIP
   ```

2. Permit the user ID:

   a. Create a BPX.SUPERUSER FACILITY class profile:

      ```
      RDEFINE FACILITY BPX.SUPERUSER
      ```

   b. If this is the first class profile, activate the FACILITY class:

      ```
      SETROPTS CLASSACT(FACILITY) SETROPTS RACLIST(FACILITY)
      ```

   c. Permit the user to the class:

      ```
      ALTUSER TCPIP OMVS(UID(23) PROGRAM ('/bin/sh') HOME('/'))
      PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(TCPIP) ACCESS(READ)
      ```

      In this example, the user ID is TN3270 and the UID is 23. The UID can be any nonzero number. UID 23 was used to match the well-known Telnet port number.

   d. Refresh the FACILITY class:

      ```
      SETROPTS RACLIST(FACILITY) REFRESH
      ```

Consider program properties table attributes for the TN3270E server

The MVS default program properties table (PPT) has the TN3270 module set up as privileged, non-swappable, non-cancelable, running in key 6, and system task. These settings give TN3270 the same priority as the TCP/IP stack. Either the privileged or system task designation causes the started job to be assigned to the SYSSTC service class. The priority can be changed by assigning the job name to another service class within the STC subsystem.

Note: The default PPT entry sets the Telnet server to non-cancelable. As a non-cancelable application, the TN3270E server should not be started automatically by a TCP/IP stack using the AUTOLOG function. If the TCP/IP stack is recycled, the following messages are issued repeatedly:

```
EZZ0621I AUTOLOG FORCING TN3270B, REASON: AUTOLOGGING SCHEDULED
IEE838I TN3270B NON-CANCELABLE - ISSUE FORCE ARM
```

To prevent this, specify NOAUTOLOG on the PORT reservation statement as follows:

```
PORT 23 TCP TN3270B NOAUTOLOG
```

Customize the JCL for the TN3270 started task

You can find a sample JCL in SEZAINST(EZBTNPRC). The only valid parameter that can be passed in from the JCL is the component trace options parmlib member name.

Specify the customized profile data set name on the PROFILE DD entry in the JCL. The data set must be fixed and blocked with a record length between 56 and 256. The block size must be evenly divisible by the record length. Normally, the profile member is placed into a PDS that has a record length of 80, and blocked accordingly.

Specify the customized tcpdata data set name on the //SYSTCPD DD statement in the JCL.

The JCL for the started task is shown in Example 2-6. Notice the use of system symbolics.

Example 2-6 JCL for the TN3270E server: TN3270

```
BROWSE    SYS1.PROCLIB(TN3270B) - 01.00             Line 00000000 Col 001 080
//TN3270B   PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=TELNB31A,TCPDATA=DATAB&SYSCLONE.
//TN3270B  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
```

2.2.3 Activation of the TN3270E server

To start the TN3270E started task, issue the MVS START command or automate it with an automation package:

```
S TN3270B
```

The initialization messages are shown in Example 2-7.

Example 2-7 Telnet server initialization message

```
S TN3270B
$HASP100 TN3270B  ON STCINRDR
IEF695I START TN3270B  WITH JOBNAME TN3270B  IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TN3270B  STARTED
IEF196I IEF237I DC63 ALLOCATED TO SYS00109
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TN3270B SERVER STARTED
EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 387
            TCPIPB.TCPPARMS(TELNB31A)
EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
            TCPIPB.TCPPARMS(TELNB31A)
IEF196I IEF285I   TCPIP.SEZALOAD                                KEPT
IEF196I IEF285I   VOL SER NOS= Z1CRB1.
EZZ6041I TN3270B SNMP SUBAGENT INITIALIZATION COMPLETE
EZZ6003I TN3270B LISTENING ON PORT 23
```

Note: TN3270B uses non-reusable address spaces. For information about how to start TN3270E by using a reusable address space ID (REUSASID), see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996.

2.2.4 Verification of the TN3270E server

For all commands referred to in this section, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775 for detailed command usage and *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781 for detailed command syntax.

This section discusses the following topics:

- Use HELP for Telnet commands
- Use OBEYFILE to modify a configuration
- Use OBEYFILE and TESTMODE to syntax check the profile statements
- Check maximum connections supported
- Check the total number of TN3270 ports defined in the profile
- Use DISPLAY commands to view the status of TN3270 resources
- Use display PROFILE to view profile information
- Use display CLIENTID command to view client ID information
- Use display OBJECT command to view object information
- Use display CONN command to view connection information

Use HELP for Telnet commands

Telnet has created its own help commands and STOR command to replace the TCP/IP versions. Any Telnet command that is submitted to the TCP/IP address space is ignored, and the message EZZ0210I is issued to indicate it. The syntax is as follows:

```
D TCPIP,tnproc,HElp
D TCPIP,tnproc,HElp,STOR
D TCPIP,tnproc,HElp,Telnet
 <CLientID | CONNection | INACTLUS | OBJect | PROFile>
D TCPIP,tnproc,HElp,LUNS
 <INACTLUS | OBJect>
D TCPIP,tnproc,HElp,XCF
V TCPIP,tnproc,HElp
V TCPIP,tnproc,HElp,Obeyfile
V TCPIP,tnproc,HElp,Telnet
 <ABENDTRAP | ACT | DEBug | INACT | QUIesce | RESUME | STOp>
V TCPIP,tnproc,HElp,LUNS
```

Telnet help commands are used to see the format of any Telnet DISPLAY or VARY command. Each help command shows the options that are available at the next level of detail. For example, if you issue `D TCPIP,tnproc,HELP`, you see that you can issue help for either STOR, TELNET, LUNS, or XCF. If you then issue `D TCPIP,tnproc,HELP,TELNET`, you see all the display Telnet options that are available.

The Telnet STOR command displays the current and maximum storage usage status and identifies the service level of Telnet modules in the previous release, as shown in Example 2-8.

Example 2-8 Telnet server storage usage

```
D TCPIP,TN3270B,STOR
EZZ8453I TELNET STORAGE 578
EZZ8454I TN3270B  STORAGE         CURRENT   MAXIMUM     LIMIT
EZZ8455I TN3270B  ECSA               85K       85K   NOLIMIT
EZZ8455I TN3270B  POOL             5706K     5708K   NOLIMIT
EZZ8455I TN3270B  64-BIT COMMON       0M        0M   NOLIMIT
EZZ8455I TN3270B  CTRACE          262372K   262372K   262372K
EZZ8459I DISPLAY TELNET STOR COMPLETED SUCCESSFULLY
```

Contrasted with the TCPIP STOR command output, the Telnet STOR command displays the storage usage status for CTRACE. TCPIP CTRACE storage resides in a separate data space and is not part of the storage display. However, Telnet CTRACE resides in Telnet's private address space. If only the total POOL value is shown, the CTRACE amount obscures the amount of storage that is used by Telnet processes. Therefore, before the data is presented,

the CTRACE amount is subtracted from the total POOL amount and presented on its own line.

The CTRACE amount appears large because Telnet always allocates a 256 M block of storage to support the largest CTRACE BUFSIZE amount. This storage is not wrapped until it is filled in with data. You will never use real storage resources for more than the amount you define on the CTRACE BUFSIZE parameter.

When you forget to state the Telnet server procedure name on the `V TCPIP,tnproc,OBEYFILE` command to update your profile, the EZZ0209I message is issued to indicate that the Telnet server configuration statements are ignored in TCP/IP instead of being processed by the default TCP/IP stack.

Note: When you migrate your TN3270E server from running within your stack task to its own address space, you will probably need to update your operational procedures and automation.

Use OBEYFILE to modify a configuration

The OBEYFILE command is useful when modifying the TN3270E server configuration dynamically. You need to understand, however, how the server profile configuration is processed when it is updated by the OBEYFILE command.

Important: When using the VARY TCPIP,tnproc,OBEYFILE command to update the TN3270 configuration, new profile statements *completely replace the profile statements* that are in use before the update. For a successful port update, both TELNETPARMS and BEGINVTAM blocks are required for each port started or modified. The updates are *not* cumulative from the previous profile. If only one change is needed in the new profile, change the old profile or copy the profile to another data set member and make the change using the OBEYFILE command.

After VARY TCPIP,,OBEYFILE command processing completes, the new profile is labeled the current profile, and the replaced profile becomes profile 0001. After another update, the new update becomes the current profile and the replaced profile becomes profile 0002. If the profile update is for a subset of the active ports, the ports not being updated remain unchanged. Profile debug messages can be suppressed by coding DEBUG OFF or DEBUG SUMMARY in TELNETGLOBALS and placing it before all other Telnet statement blocks.

New connections are associated with the current profile and use the mappings and parameters defined by that profile. Even if a VARY TCPIP,,OBEYFILE command updates the port, existing connections remain associated with the same profile. The statements of non-current profiles remain in effect and continue to support all connections that were established when the non-current profile was the CURRent profile. When all connections associated with a non-current profile end, the parameter and mapping structure storage for the non-current profile is released, leaving only a small anchor block that is used for profile displays. At this point the profile is considered INACTIVE.

Managing non-current profiles

Preexisting profiles often have connections still associated with them even when there is no SNA LU-LU session associated with them. For example, a USSMSG10 display or a Telnet solicitor prompt represents an active TCP connection but no underlying SNA session. Frequent profile updates to the TN3270 profile produce additional non-current profiles with connections to TN3270 but not to an SNA session behind the TN3270 server. Such non-productive connections can consume significant storage. Such profile structures cannot be released because a TCP connection is still active.

You can better manage this release of unproductive storage by implementing the parameter ProfileInactive in the TelnetGlobals, TelnetParms, or Parmsgroup statement. If the default of 1800 seconds is used, connections using noncurrent profiles will be dropped after being without a SNA session for at least 30 minutes. **ProfileInactive** controls how long a connection can stay connected without an SNA session when associated with a non-current profile.

Use OBEYFILE and TESTMODE to syntax check the profile statements

To validate a TN3270 profile without applying the profile, specify TESTMODE (a TELNETPARMS-only parameter). When no errors are reported, remove the TESTMODE statement. A sample of the TESTMODE statement is shown in Example 2-9.

Example 2-9 TESTMODE statement in TN3270 profile

```
TELNETPARMS
   . . .
   TESTMODE
   . . .
ENDTELNETPARMS
```

Use the OBEYFILE command to validate the new profile and also to activate the new profile. The OBEYFILE command is entered the same either way. The TESTMODE statement makes the difference between validation and actual activation. The OBEYFILE command is shown in Example 2-10.

Example 2-10 OBEYFILE command to validate or activate a TN3270 profile

```
V TCPIP,TN3270B,OBEYFILE,TCPIPB.TCPPARMS(TELNB31A)
```

Example 2-11 shows the resulting messages from the OBEYFILE command when the TESTMODE statement is *included* in the TN3270 profile.

Example 2-11 Messages issued when TESTMODE is included in the TN3270 profile

```
V TCPIP,TN3270B,OBEYFILE,TCPIPB.TCPPARMS(TELNB31A)
EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 610
            TCPIPB.TCPPARMS(TELNB31A)
EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
            TCPIPB.TCPPARMS(TELNB31A)
EZZ6018I TN3270B PROFILE TESTMODE COMPLETE FOR PORT 23      1
EZZ6038I TN3270B COMMAND OBEYFILE  COMPLETE
```

In this example, message EZZ6018I 1 indicates that the TESTMODE scan has been performed and no activation of the profile occurred.

Example 2-12 shows the resulting messages from the OBEYFILE command when the TESTMODE statement was *removed* from the TN3270 profile.

Example 2-12 Messages issued when TESTMODE is removed from the TN3270 profile

```
V TCPIP,TN3270B,OBEYFILE,TCPIPB.TCPPARMS(TELNB31A)
EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 594
            TCPIPB.TCPPARMS(TELNB31A)
EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
            TCPIPB.TCPPARMS(TELNB31A)
EZZ6018I TN3270B PROFILE UPDATE COMPLETE FOR PORT    23     1
EZZ6038I TN3270B COMMAND OBEYFILE  COMPLETE
```

In this example, message EZZ6018I 1 indicates that the profile has been activated and the TN3270 server is listening on the specified port.

Note: You can specify the TESTMODE statement in the initial startup profile. However, the end result is that no port is opened and that clients cannot connect. It is as though no profile statements existed in the initial profile.

Check maximum connections supported

For each port, TN3270 uses setrlimit() to automatically set the MaxFileProc value to the maximum allowed by z/OS UNIX, currently 131,072. Each TN3270 port will support that number of connections. Be sure that MaxSockets is large enough to support the anticipated number of sockets used by the system. Review the settings in the SYS1.PARMLIB(BPXPRMxx) member. If your system is IPv6 enabled, TN3270 listening sockets are IPv6. Be sure to set IPv6 MaxSockets appropriately.

Check the total number of TN3270 ports defined in the profile

TN3270 supports up to 255 ports on one TCP/IP stack. Make sure your configuration does not exceed that. A unique TELNETPARMS block must be created for each port or qualified port. TN3270 allows the use of the same BEGINVTAM block for all ports, some ports, or a unique BEGINVTAM block for each port. Both TELNETPARMS and BEGINVTAM blocks are required for each port started or modified by a VARY TCPIP,,OBEYFILE command. One or more PORT reservation statements can be specified. Each port should be defined with its own explicit TELNETPARMS block and its own explicit BEGINVTAM block to avoid confusion.

Use DISPLAY commands to view the status of TN3270 resources

For all commands referred to in this section, see the following resources:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for detailed command usage
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781, for detailed command syntax and examples

Use the **DISPLAY TCPIP,tnproc,comm** system command to request Telnet information.

If the stack is running in IPv6 or if the FORMAT LONG configuration statement is specified, then tabular style displays will be used in a format using a second line to display the data when a client ID appears on the line. The displays will be in the single line format if the stack is running in IPv4 and the FORMAT LONG configuration statement is not specified. To ensure uniformity in the displays, if the second line format is in effect, then any IPv4 address will be displayed on the second line even if the data would fit on a single line.

In an effort to conserve space in our examples, we used the SHORT format for all our displays.

Most of the Telnet DISPLAY commands are grouped into one of the following categories:

- **D TCPIP,tnproc,PROF,....**
- **D TCPIP,tnproc,CLID,....**
- **D TCPIP,tnproc,OBJ,....**
- **D TCPIP,tnproc,CONN,....**

Note: Profile, connection, and port-related displays contain a port description line that identifies the port for the *preceding* lines of data.

All commands containing the PROFILE= parameter are considered part of the profile group because the commands categorize (and display) the information based on what profile it is contained in. All of these commands search all profiles that match the PROFILE= search criteria. After a match is found, the other parameters are used to determine what is displayed for the profile.

Use display PROFILE to view profile information

The PROFILE display command enables you to determine what profile-wide options are in effect for each profile, which profiles are being used, and how many users are on each profile. In the next few display examples, notice the difference between summary output and detailed output. And because port 23 is defined as a BASIC (non-secure) port, any display command specifying SECURE information will result in a “no matches found” condition. Fields of interest to this discussion are *highlighted* in the output messages. Because port 23 is a BASIC port (and the only BASIC port), the results of specifying BASIC or omitting it are the same. A profile summary report shows a summary of the port settings; see Example 2-13.

Example 2-13 Display PROF, SUM shows profile summary for all profiles

```
D TCPIP,TN3270B,PROF,SUM
EZZ6060I TN3270B PROFILE DISPLAY 768
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      0
------------------------------------------------------------
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   TNSACONFIG        DISABLED
   DEBUG TASK        EXCEPTION  CONSOLE
   DEBUG CONFIG      EXCEPTION  CONSOLE
   DEBUG CONFIG      TRACEOFF
14 OF 14 RECORDS DISPLAYED
```

A profile detail report (Example 2-14) shows the port settings and includes a legend to assist in interpreting the abbreviated settings.

Example 2-14 Display PROF, DET shows profile detail for all profiles

```
D TCPIP,TN3270B,PROF,DET,MAX=*
EZZ6080I TN3270B PROFILE DISPLAY 776
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  LM-----  ---S--------  --F  -B--*----  *---ST*  S--- *TPARMS
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* CURR
 PERSISTENCE
   LUSESSIONPEND
   MSG07
   ...
 FUNCTIONS
   ...
   TN3270E
   SNAEXTENT
```

```
   ...
 DIAGNOSTICS
   DEBUG CONN        EXCEPTION  CONSOLE
   DEBUG CONN        TRACEOFF
   FULLDATATRACE
 SECURITY
   PORT                   23
   CONNTYPE          BASIC
   KEYRING           TTLS/**N/A**
   CRLLDAPSERVER     NONE/TTLS/**N/A**
   ENCRYPTION        **N/A**
   CLIENTAUTH        **N/A**
   NOEXPRESSLOGON
   NONACUSERID
   NOSSLV2
 TIMERS
   ...
 MISCELLANEOUS
   ...
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      0
------------------------------------------------------------
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   ...
```

The profile report can be limited to BASIC ports only, as shown in Example 2-15.

Example 2-15 Display PROF, BASIC, SUM shows profile summary for basic profiles

```
D TCPIP,TN3270B,PROF,PROF=BASIC,SUM
EZZ6060I TN3270B PROFILE DISPLAY 787
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      0
------------------------------------------------------------
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   TNSACONFIG        DISABLED
   DEBUG TASK        EXCEPTION  CONSOLE
   DEBUG CONFIG      EXCEPTION  CONSOLE
   DEBUG CONFIG      TRACEOFF
14 OF 14 RECORDS DISPLAYED
```

The profile report can be limited to SECURE ports only, as shown in Example 2-16 on page 60. Because we have no secure ports defined in this profile, no entries are listed.

Example 2-16 Display PROF, SECURE, SUM shows profile summary for secure profiles

```
D TCPIP,TN3270B,PROF,PROF=SECURE,SUM
EZZ6060I TN3270B PROFILE DISPLAY 794
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   TNSACONFIG        DISABLED
   DEBUG TASK        EXCEPTION  CONSOLE
   DEBUG CONFIG      EXCEPTION  CONSOLE
   DEBUG CONFIG      TRACEOFF
8 OF 8 RECORDS DISPLAYED
```

Use display CLIENTID command to view client ID information

The CLIENTID display can be used to see what client IDs are defined in the profile and details about the client IDs. In the next few display examples, notice the difference between summary output and detailed output. And because port 23 is defined as a BASIC (non-secure) port, any display command specifying SECURE information will result in a “no matches found” condition. Fields of interest to this discussion are *highlighted* in the output messages. Because port 23 is a BASIC port (and the only BASIC port), the results of specifying BASIC or omitting it are the same. A ClientID summary report shows any client groups you might have defined, as shown in Example 2-17.

Example 2-17 Display CLID, SUM shows clientID summary for all profiles

```
D TCPIP,TN3270B,CLID,SUM
EZZ6082I TN3270B CLIENTID LIST 796
USERID
  NO CLIENT IDS
HOSTNAME
  NO CLIENT IDS
IPADDR
  NO CLIENT IDS
USERGRP
  NO CLIENT IDS
HNGRP
  NO CLIENT IDS
IPGRP
  NO CLIENT IDS
DESTIP
  NO CLIENT IDS
LINKNAME
  NO CLIENT IDS
DESTIPGRP
  NO CLIENT IDS
LINKGRP
  NO CLIENT IDS
NULL
  NULL
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
26 OF 26 RECORDS DISPLAYED
```

All CLIENTID types are listed for reference. If a CLIENTID type is not defined, then “NO CLIENT IDS” are indicated.

A ClientID detail report shows the how many connections are associated with which client ID groups, as shown in Example 2-18.

Example 2-18 Display CLID, DET shows clientID detail for all profiles

```
D TCPIP,TN3270B,CLID,DET
EZZ6081I TN3270B CLIENTID DISPLAY 802
CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
------------------ ------ --------- -------- ---------- --------
NULL
  NULL
                        1 DEFAPPL   SC31TS               --------
----- PORT:    23  ACTIVE            PROF: CURR CONNS:      1
------------------------------------------------------------
10 OF 10 RECORDS DISPLAYED
```

The ClientID report can be limited to BASIC ports only, as shown in Example 2-19.

Example 2-19 Display CLID, BASIC, SUM shows clientID summary for basic profiles

```
D TCPIP,TN3270B,CLID,PROF=BASIC,SUM
EZZ6082I TN3270B CLIENTID LIST 808
USERID
  NO CLIENT IDS
HOSTNAME
  NO CLIENT IDS
IPADDR
  NO CLIENT IDS
USERGRP
  NO CLIENT IDS
HNGRP
  NO CLIENT IDS
IPGRP
  NO CLIENT IDS
DESTIP
  NO CLIENT IDS
LINKNAME
  NO CLIENT IDS
DESTIPGRP
  NO CLIENT IDS
LINKGRP
  NO CLIENT IDS
NULL
  NULL
----- PORT:    23  ACTIVE            PROF: CURR CONNS:      1
------------------------------------------------------------
26 OF 26 RECORDS DISPLAYED
```

The ClientID report can be limited to SECURE ports only. Because there are no SECURE ports defined in the profile, message EZZ6057I indicates there are no entries to list, as shown in Example 2-20.

Example 2-20 Display CLID, SECURE, SUM shows client ID summary for secure profiles

```
D TCPIP,TN3270B,CLID,PROF=SECURE,SUM
EZZ6082I TN3270B CLIENTID LIST 813
EZZ6057I NO QUALIFYING MATCHES
3 OF 3 RECORDS DISPLAYED
```

Use display OBJECT command to view object information

The OBJECT display can be used to see what objects are defined in the profile and some details about the objects. In the next few display examples, notice the difference between summary output and detailed output. And because port 23 is defined as a BASIC (non-secure) port, any display command specifying SECURE information will result in a “no matches found” condition. Fields of interest to this discussion are *highlighted* in the output messages. Because port 23 is a BASIC port (and the only BASIC port), the results of specifying BASIC or omitting it are the same.

The object summary report shows any objects that might be defined in the profile and what their assignments are, as shown in Example 2-21.

Example 2-21 Display OBJ, SUM shows object summary for all profiles

```
D TCPIP,TN3270B,OBJ,SUM
EZZ6084I TN3270B OBJECT LIST 819
ARAPPL
  SC*       NVAS*     TSO*      *
DEFAPPL
  SC31TS
PRTAPPL
  NO OBJECTS
LINEAPPL
  NO OBJECTS
MAPAPPL
  NO OBJECTS
USS
  NO OBJECTS
INT
  NO OBJECTS
LU
  NO OBJECTS
LUGRP
  *DEFLUS*
SLUGRP
  NO OBJECTS
APPLLUG
  NO OBJECTS
PRT
  NO OBJECTS
PRTGRP
  NO OBJECTS
```

```
SPRTGRP
  NO OBJECTS
PARMSGRP
  *DEFAULT  *TGLOBAL  *TPARMS
MONGRP
  NO OBJECTS
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
36 OF 36 RECORDS DISPLAYED
```

All OBJECT types are listed for reference. If an OBJECT type is not defined, then NO OBJECTS is indicated.

The object detail report indicates how many connections there are for each object group, as shown in Example 2-22.

Example 2-22 Display OBJ, DET shows object detail for all profiles

```
D TCPIP,TN3270B,OBJ,DET
EZZ6083I TN3270B OBJECT DISPLAY 821
OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
----------  ------ --------- ---------------- ---------- --------
ARAPPL
 SC*             1
                                                        -A------
 NVAS*           0
                                                        -A------
                                                      5 ----Q---
 TSO*            0
                                                        -A-D----
 *               0
                                                        -A------
DEFAPPL          0
                                                        -I------
DEFAPPL
 SC31TS          1 NULL      NULL
                                                        --------
LUGRP
 *DEFLUS*        1
                                                        --------
PARMSGRP
 *DEFAULT          -------NO MAPPING---------
                                                        --------
 *TGLOBAL          -------NO MAPPING---------
                                                        --------
 *TPARMS           -------NO MAPPING---------
                                                        --------
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
32 OF 32 RECORDS DISPLAYED
```

The object report can be limited to basic or to secure ports. Because there are no secure ports defined in this profile, message EZZ6057I indicates there are not entries to list, as shown in Example 2-23.

Example 2-23 Display OBJ, SECURE, SUM shows object summary for secure profiles

```
D TCPIP,TN3270B,OBJ,PROF=SECURE,SUM
EZZ6084I TN3270B OBJECT LIST 825
EZZ6057I NO QUALIFYING MATCHES
3 OF 3 RECORDS DISPLAYED
```

Use display CONN command to view connection information

The CONNECTION display command without the CONN= parameter gives you a high-level view of what connections exist and what they are being used for. A connection report shows all existing connections to each active port, as shown in Example 2-24.

Example 2-24 Display CONN shows all connections to the Telnet server

```
D TCPIP,TN3270B,CONN,MAX=*
EZZ6064I TN3270B CONN DISPLAY 852
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000347    ::FFFF:10.1.100.221..4271
                                   SC31BB02 SC31TS03  TAE SNX32702
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
9 OF 9 RECORDS DISPLAYED
```

The connection report can be limited to basic ports only, showing only basic connections, as shown in Example 2-25.

Example 2-25 Display CONN, BASIC shows the basic connections only

```
D TCPIP,TN3270B,CONN,PROF=BASIC,MAX=*
EZZ6064I TN3270B CONN DISPLAY 855
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000347    ::FFFF:10.1.100.221..4271
                                   SC31BB02 SC31TS03  TAE SNX32702
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
9 OF 9 RECORDS DISPLAYED
```

The connection report shows secure ports and secure connections, as shown in Example 2-26.

Example 2-26 Display CONN, SECURE shows the secure connections

```
D TCPIP,TN3270B,CONN,PROF=SECURE,MAX=*
EZZ6064I TN3270B CONN DISPLAY 859
EZZ6057I NO QUALIFYING MATCHES
3 OF 3 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and SUM parameter gives a *summary* look at one single connection, as shown in Example 2-27.

Example 2-27 Display CONN, CONN=, SUM shows summary information for one connection only

```
D TCPIP,TN3270B,CONN,CONN=00000347,SUM
EZZ6064I TN3270B CONN DISPLAY 862
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000347    ::FFFF:10.1.100.221..4271
                                  SC31BB02 SC31TS03  TAE SNX32702
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
9 OF 9 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and DET parameter gives a *complete* look at one single connection, as shown in Example 2-28.

Example 2-28 Display CONN, CONN=, DET shows detail information for one connection only

```
D TCPIP,TN3270B,CONN,CONN=00000347,DET
EZZ6065I TN3270B CONN DISPLAY 877
  CONNECTED: 17:11:14  09/28/2010  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 00000347 SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.221..4271
    DESTIP..PORT: ::FFFF:10.1.1.20..23
    LINKNAME: VIPA1L
  PORT:    23 QUAL: NONE
    AFFINITY: TCPIPB
    STATUS: ACTIVE   BASIC
    ACCESS: NON-SECURE
  PROTOCOL: TN3270E           DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
    OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                       NEWENV FUNCTIONS: --
  LUNAME: SC31BB03
  APPL: SC31TS01
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
  MAPPING TYPE:  CONN IDENTIFIER
                       OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  NL (NULL)
                      >*DEFLUS*                   --------
    DEFLT APPL: NL (NULL)
                       SC31TS                     --------
    USS TABLE:  **N/A**
    INT TABLE:  **N/A**
```

```
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  LM-----  ---S--------  --F  -B--*----  *---ST*  S--- *TPARMS
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* TP-CURR
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* <-FINAL
38 OF 38 RECORDS DISPLAYED
```

The CONNECTION display command with the LUNAME= parameter and SUM parameter gives a summary look at the connection assigned to that LU name, as shown in Example 2-29.

Example 2-29 Display CONN, LUNAME=, SUM shows summary information for one LU name only

```
D TCPIP,TN3270B,CONN,LUNAME=SC31BB03,SUM
EZZ6064I TN3270B CONN DISPLAY 883
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000347    ::FFFF:10.1.100.221..4271
                                   SC31BB03 SC31TS01  TAE SNX32702
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
9 OF 9 RECORDS DISPLAYED
```

The CONNECTION display command with the LUNAME= parameter and DET parameter gives you a complete look at the connection assigned to that LU name. Because our single example is only one connection, using either the CONN= or the LUNAME= parameter results in displaying the same connection.

2.2.5 Administration and management of the TN3270E server

The following tools can be useful when managing the TN3270E server environment:

- Use VARY to quiesce, resume, and stop a TN3270 port
- Use VARY to change the status of TN3270 LUs
- Check client connection status
- LU name assignment user exit
- System symbols for unformatted system services tables
- Queued session timer
- Performance monitoring data collection
- Real-time SMF information service access control

Use VARY to quiesce, resume, and stop a TN3270 port

Telnet VARY commands enable the operator to change the state of TN3270 ports, and enable or disable the use of certain TN3270 ports. These commands are as follows:

- VARY TCPIP,tnproc,QUIESCE a port to block any new connection requests but allow existing connections to continue activity.
- VARY TCPIP,tnproc,RESUME a port to end the QUIESCEd state and allow new connection requests.

- VARY TCPIP,tnproc,STOP a port to end connections on the port, and close the port, and discard all information for that port as though it were never defined.
- VARY TCPIP,tnproc,OBEYFILE to start, restart, or change a port by updating the TN3270 profile.

The VARY TCPIP,tnproc,STOP and VARY TCPIP,tnproc,OBEYFILE commands can be used to stop a TN3270 port and then restart that port or a new port without stopping the TN3270 server started task.

The QUIESCE command is shown in Example 2-30.

Example 2-30 QUIESCE Port 23

```
V TCPIP,TN3270B,QUIESCE,PORT=23
EZZ6038I TN3270B COMMAND QUIESCE 23 COMPLETE
EZZ6003I TN3270B QUIESCED ON PORT    23
```

To verify the QUIESCE command, use a display profile command as shown in Example 2-31.

Example 2-31 Port status after QUIESCE command

```
D TCPIP,TN3270B,PROF
EZZ6060I TN3270B PROFILE DISPLAY 898
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
----- PORT:    23  QUIESCED         PROF: CURR CONNS:      1
------------------------------------------------------------
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   TNSACONFIG        DISABLED
   DEBUG TASK        EXCEPTION  CONSOLE
   DEBUG CONFIG      EXCEPTION  CONSOLE
   DEBUG CONFIG      TRACEOFF
14 OF 14 RECORDS DISPLAYED
```

If a client attempts to connect while the port is quiesced, the request is rejected, as shown in Example 2-32. We directed a connection request toward the quiesced port using the TSO TELNET command.

Example 2-32 Connection request is rejected when port is quiesced

```
===> TELNET 10.1.1.20
EZA8200I MVS TCP/IP TELNET CS V1R13
EZA8256I Connecting to 10.1.1.20, port TELNET (23)
EZA8262I Foreign host rejected the open attempt (8547)
***
```

The RESUME command is shown in Example 2-33.

Example 2-33 RESUME port 23

```
V TCPIP,TN3270B,RESUME,PORT=23
EZZ6038I TN3270B COMMAND RESUME 23 COMPLETE
EZZ6003I TN3270B RESUMED ON PORT    23
```

To verify the RESUME command, a display profile command is shown (Example 2-34.

Example 2-34 Port status after RESUME command

```
D TCPIP,TN3270B,PROF
EZZ6060I TN3270B PROFILE DISPLAY 921
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   TNSACONFIG        DISABLED
   DEBUG TASK        EXCEPTION  CONSOLE
   DEBUG CONFIG      EXCEPTION  CONSOLE
   DEBUG CONFIG      TRACEOFF
14 OF 14 RECORDS DISPLAYED
```

An example of the STOP command is shown in Example 2-35. The client connection that was active is forced off and messages show the adverse affect on the connection. All connections on port 23 would be terminated.

Example 2-35 STOP port 23

```
V TCPIP,TN3270B,STOP,PORT=23
EZZ6038I TN3270B COMMAND STOP 23 COMPLETE
IKT100I USERID           CANCELED DUE TO UNCONDITIONAL LOGOFF
IKT122I IPADDR..PORT 10.1.100.221..4319
EZZ6010I TN3270B SERVER ENDED FOR PORT    23
```

To verify the STOP command, a display profile command is shown in Example 2-36.

Example 2-36 Port status after STOP command

```
D TCPIP,TN3270B,PROF
EZZ6060I TN3270B PROFILE DISPLAY 931
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   TNSACONFIG        DISABLED
   DEBUG TASK        EXCEPTION  CONSOLE
   DEBUG CONFIG      EXCEPTION  CONSOLE
   DEBUG CONFIG      TRACEOFF
8 OF 8 RECORDS DISPLAYED
```

Using the OBEYFILE command to restart port 23 is shown in Example 2-37.

Example 2-37 Restart port 23 with OBEYFILE command

```
V TCPIP,TN3270B,OBEYFILE,TCPIPB.TCPPARMS(TELNB31A)
EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 933
            TCPIPB.TCPPARMS(TELNB31A)
EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
            TCPIPB.TCPPARMS(TELNB31A)
EZZ6003I TN3270B LISTENING ON PORT    23
EZZ6038I TN3270B COMMAND OBEYFILE  COMPLETE
```

To verify that port 23 is again defined and active, a display profile command is shown (Example 2-38).

Example 2-38 Port status after restart with OBEYFILE command

```
D TCPIP,TN3270B,PROF
EZZ6060I TN3270B PROFILE DISPLAY 939
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
   FORMAT            LONG
   TCPIPJOBNAME      TCPIPB
   TNSACONFIG        DISABLED
   DEBUG TASK        EXCEPTION  CONSOLE
   DEBUG CONFIG      EXCEPTION  CONSOLE
   DEBUG CONFIG      TRACEOFF
14 OF 14 RECORDS DISPLAYED
```

Use VARY to change the status of TN3270 LUs

VARY TCPIP,tnproc,ACT and VARY TCPIP,tnproc,INACT LUs are for use by the Telnet server. 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.

To show a list of inactive TN3270 LUs before any are inactivated, use the Display INACTLUS command as shown in Example 2-39.

Example 2-39 Display inactive TN3270 LUs before an INACT

```
D TCPIP,TN3270B,INACTLUS
EZZ6061I TN3270B INACTLUS DISPLAY 947
EZZ6057I NO QUALIFYING MATCHES
3 OF 3 RECORDS DISPLAYED
```

An example of the INACT command to inactivate a TN3270 LU is shown in Example 2-40.

Example 2-40 INACT command to inactivate a TN3270 LU

```
V TCPIP,TN3270B,INACT,SC31BB26
EZZ6038I TN3270B COMMAND INACT SC31BB26 COMPLETE
```

To show a list of inactive TN3270 LUs, use the Display INACTLUS, command as shown in Example 2-41.

Example 2-41 Display inactive TN3270 LUs after an INACT

```
D TCPIP,TN3270B,INACTLUS
EZZ6061I TN3270B INACTLUS DISPLAY 953
INACTIVE LUS
               SC31BB26
4 OF 4 RECORDS DISPLAYED
```

The ACT command to activate a TN3270 LU is shown in Example 2-42.

Example 2-42 ACT command to activate a TN3270 LU

```
V TCPIP,TN3270B,ACT,SC31BB26
EZZ6038I TN3270B COMMAND ACT SC31BB26 COMPLETE
```

To show a list of inactive TN3270 LUs, use the Display INACTLUS command, as shown in Example 2-43. Because we reactivated SC31BB26, it no longer shows as an inactive LU.

Example 2-43 Display inactive TN3270 LUs after an ACT

```
D TCPIP,TN3270B,INACTLUS
EZZ6061I TN3270B INACTLUS DISPLAY 958
EZZ6057I NO QUALIFYING MATCHES
3 OF 3 RECORDS DISPLAYED
```

Check client connection status

If the Telnet server receives a new connection from a client IP address that already has one or more existing connections, the server checks the existing connections to make sure that they are still available. If the connections are not available, the previously existing connections are cleaned up immediately. This clean up improves the situation where a user thinks a session has failed and starts a new session (see Figure 2-5).

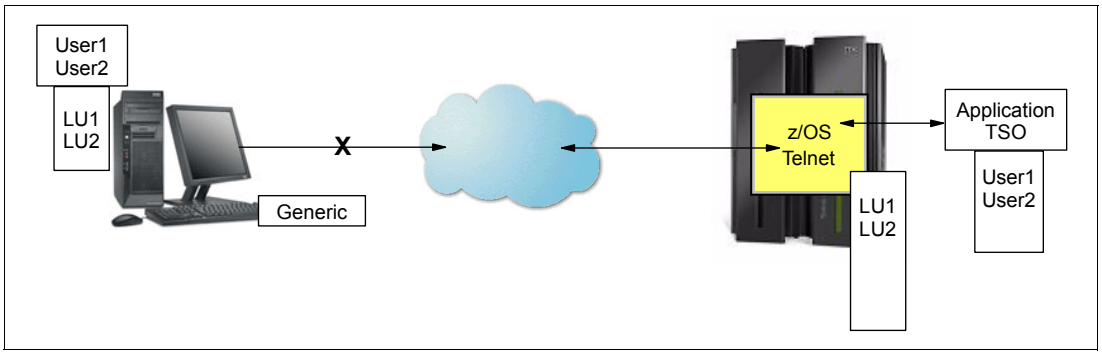


Figure 2-5 Client connection status

The *check client connection* function sends a TIMEMARK value to every pre-existing connection that is associated with the client identifier of the new connection that is being established. If a response is not received, the connection is ended. This is useful when you have a generic request configuration with many clients on a single workstation. Neither specific nor generic takeover will end an existing SNA session. An alternate method, using the *keepalive* function with ScanInterval and Timemark, creates high overhead. The check client connection method has much less overhead.

The CheckClientConn statement has the following syntax:

```
CheckClientConn sec,maxconn
```

In this command:

- The *sec* parameter specifies the number of seconds Telnet waits for clients to respond.
- The *maxconn* parameter specifies the maximum number of connections checked for a single client identifier. The default for *maxconn* is 50. You can exclude connections with parmsgroup/parmsmap statements.

Tip: This parameter is important if you are using a proxy server. A proxy server causes all client connections to appear as though they were coming from the same client IP address. A large *maxconn* setting might be needed in this situation.

The NoCheckClientConn statement has no parameters and is used to turn off CheckClientConn for specific cases.

Example 2-44 shows the definition for clients with a maximum of 60 Telnet connections.

Example 2-44 Example CheckClientConn

```
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
    CheckClientConn 5,60
ENDTELNETPARMS
```

Example 2-45 displays the profile *before* adding the CheckClientConn statement. The default setting is indicated.

Example 2-45 Telnet profile without CheckClientConn enabled

```
D TCPIP,TN3270B,PROF,DETAIL
EZZ6080I TN3270B PROFILE DISPLAY 962
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  LM-----  ---S--------  --F  -B--*----  *---ST*  S--- *TPARMS
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* CURR
 PERSISTENCE
   LUSESSIONPEND
   MSG07
   NOTKOSPECLU
   NOTKOGENLU
   NOCHECKCLIENTCONN
   NODROPASSOCPRINTER
   KEEPLU                  0 (OFF)
 FUNCTIONS
   NOOLDSOLICITOR
...
```

Example 2-46 displays the profile *after* adding the CheckClientConn statement. The default setting is indicated.

Example 2-46 Telnet profile with the CheckClientConn enabled

```
D TCPIP,TN3270B,PROF,DETAIL
EZZ6080I TN3270B PROFILE DISPLAY 972
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  LM--C--  ---S--------  --F  -B--*----  *---ST*  S--- *TPARMS 1
  LM**C**  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* CURR
 PERSISTENCE
   LUSESSIONPEND
   MSG07
   NOTKOSPECLU
   NOTKOGENLU
   CHECKCLIENTCONN          5,60
   NODROPASSOCPRINTER
   KEEPLU                   0 (OFF)
 FUNCTIONS
   NOOLDSOLICITOR
...
```

In this example, the parameter was set in the TelnetParms statement 1.

Important: Use CheckClientConn in place of setting low Scaninterval and Timemark values. ScanInterval and Timemark is intended for connection cleanup, not connection recovery.

LU name assignment user exit

Most LU assignment requirements can be satisfied using the Telnet LU group and LU mapping statements. This way, fixed IP addresses or hostnames allow mapping of specific unformatted system services (USS) tables to specific users. The LU name and the USS table are often functionally linked together and both must be mapped to the same client identifiers. However, there are cases when the LU assignment requirements are so specific that Telnet cannot satisfy them. In these cases, the LU name assignment user exit might be the solution. Some customers use an LU exit to map LU names to nonstandard client identifiers, or there might be so many exceptions to a client identifier range or group that an LU exit table is easier to maintain. The support for the USS table assignment from the LU exit routine provides space in the parameter list to return the USS table names in the following formats:

- 3270 format USS table
- SCS format USS table
- Interpret table

The LU exit assigned USS table takes precedence over a mapped USS table.

Important: An unformatted system services assignment for a TN3270E connection must be without SimClientLU. The LU name must be assigned during TN negotiation before the first USSMSG10 screen is sent. Non-TN3270E connections are not assigned an LU name until the application name is chosen. By then, the first USSMSG10 screen has already been sent to the client.

Make sure that any LU exit that assigns USS table names is used only on V1R8 and above. The parameter list has been expanded to accommodate the USS table names. Attempting to write these names into a down-level parameter list will result in storage overlays.

Existing functions are not changed by this function. You must change your LU exit to use this function.

Important: To keep storage usage low in your system, minimize the number of unique USS tables loaded, the number of unique 3270/SCS pairs created, and the number of active profiles.

System symbols for unformatted system services tables

In prior releases it was possible to use symbolic variables, such as the LU name, the IP addresses, the IP port, and the IP host name in USS message, especially message 10, definitions. This has been expanded to include system symbols, such as the system name, system release, and others.

An easy way to review potential symbols is with the MVS command **D SYMBOLS** to display the potential symbols that might be used in MSG10. Example 2-47 illustrates usage of system symbols.

Example 2-47 USS table source code with SYSNAME and SYSR11

```
 MSG10    DC      AL2(MSG10E-MSG10S)
 MSG10S   EQU     *
          DC      X'F5C31140401D40' ERASE/WRITE,WCC,SBA R1C1,
          DC C'WTSC30 You are connected to a non-SNA terminal - - - '
          DC      X'11C1501DE4'
          DC CL50'                                                  '
          DC CL30'                                '
          DC CL50'System Name:  &&SYSNAME.                          '
          DC CL30'z/OS Release: &&SYSR11.          '
          DC CL50'                                                  '
          DC CL30'                                '
.....................................
```

Note: Notice the double ampersand (&) on the symbolic name. This notation is necessary.

This function can be useful for diagnostic information; for example, including the LPAR name can be helpful in many situations. The system symbol support is not present in the native VTAM USS support. Any system symbol coded on a shared table is not converted by VTAM. The symbol name is displayed if the table is used for VTAM USS processing.

Figure 2-6 shows the USS table hello screen presentation from the definitions that we used in Example 2-47 on page 73.

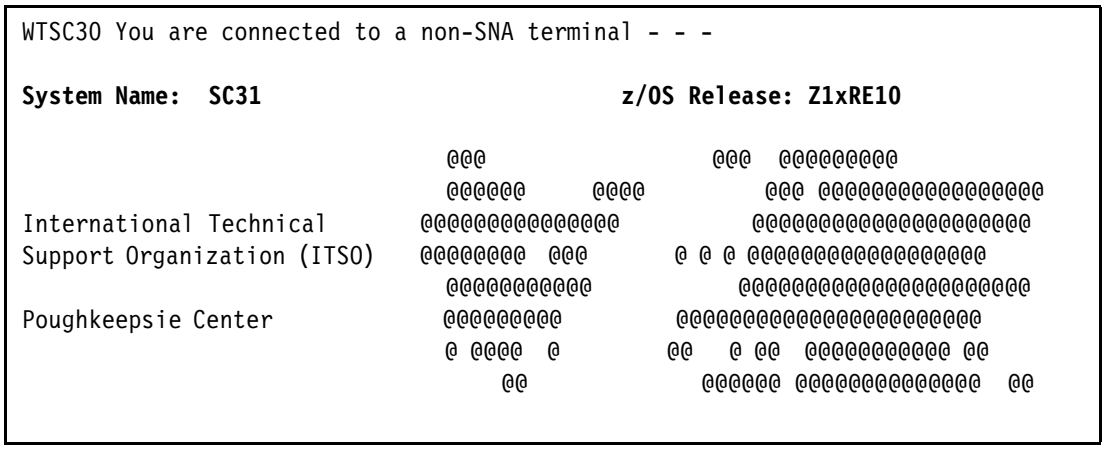


Figure 2-6 The USSTEST2 compiled including SYSNAME and SYSR9

Example 2-48 shows the corresponding definition in the TCP/IP profile.

Example 2-48 TCP/IP profile mapping the new USS table

```
USSTCP USSTEST1                1
USSTCP USSTEST2 10.1.100.222   2
```

In this example, the numbers correspond to the following information:

1. USS table for general purposes
2. USS table mapped for a specific IP address

Queued session timer

Logon manager applications are popular and usually operate as a default application that sends a selection screen to the user. After the user specifies the destination application choice, the logon manager issues a CLSDST macro with OPTCD=PASS to the destination application. When logging off the destination application, if QSession is not specified, Telnet redrives the session to the initial setup which means to close the LU ACB and sends the USSMSG10 to the screen. When QSession is specified, Telnet keeps the LU ACB open and does not redrive the session. Telnet assumes, based on QSession, that a BIND will arrive from the session manager application. During this time, the user's keyboard locks up and no new commands can be entered.

We can specify how long to wait after receiving an UNBIND. This allows TN3270 to redrive setup if a session manager does not bind within a specified time after the previous session's unbind. It eliminates the need for the user to disconnect/reconnect in some error cases. QSession is the associated parameter on the RESTRICTAPPL or ALLOWAPPL statements.

Example 2-49 Display to see options for Allow/Restrict Appls

```
D TCPIP,TN3270B,OBJ,TYPE=ARAPPL
EZZ6083I TN3270B OBJECT DISPLAY 990
OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
----------  ------ --------- ---------------- ---------- --------
ARAPPL
 SC*             0                                        -A------
 NVAS*           0                                        -A------      1
                                                        5 ----Q---      2
 TSO*            0                                        -A-D----
 *               0                                        -A------
DEFAPPL          0                                        -I------
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      0
-------------------------------------------------------------
19 OF 19 RECORDS DISPLAYED
```

The numbers in this example correspond to the following information:

1. The **A** in the OPTIONS field indicates AllowAppl.
2. The **Q** in the OPTIONS field indicates QSession.

   The **5** in the ITEM SPECIFIC field is the time value. The definition in the TN3270E server configuration file is `ALLOWAPPL NVAS* QSESSION,5`.

Important: When using a Qsession Timer:

- A timer value must be specified.
- Set a time high enough to avoid potential conflicts where the application is sending a BIND at the same time Telnet is cleaning up and redriving the initial setup.
- Do not set the value so high that the user appears to see a hung terminal and manually terminates the session.

Performance monitoring data collection

TN3270E server performance data was difficult to collect in the early releases of z/OS. Information was provided for a specified terminal instead of for the whole server. The Network Management Interface (EZBNMIFR callable API) collects data and avoids this problem. It bypasses SNMP and calls the TN3270E server directly and returns all data in a single large data block. The same data is reported by EZBNMIFR as is reported with SNMP, but EZBNMIFR is more efficient when examining all Telnet sessions.

We encourage you to use the Telnet SMF SNA termination record to collect “Life of Session” data. Multiple SNA sessions are possible during the Life of a single connection and the data is reported through Telnet SMF SNA session termination record (Type 119/subtype 21). The following items are collected:

- Transaction count
- Round trip and IP response time totals
- Sum of squares for round trip, IP and SNA
- Transaction counts by time bucket

With this data we can calculate some useful session information, including:

- Averages for round trip, IP and SNA response times
- Variance and standard deviation for round trip, IP and SNA response times

Sliding window data is not reported because data is collected at the end of each session.

The MonitorGroup and MonitorMap parameters in the BEGINVTAM block must be in place for Telnet to capture performance data.

Additional information about the Network Management Interface is available in *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787. Two request options of particular interest are:

- GetTnMonitorGroups: Obtain information about TN3270E monitor groups
- GetTnConnectionData: Obtain information about TN3270E connection performance data.

Filters for GetTnConnectionData are available, including the following:

- Resource ID, Server resource ID, Local IP address, Local IP address prefix, Local port, Remote IP address, Remote IP address prefix and Remote port
- LU Name, Monitor Group Identifier, Application Name

Telnet SMF 119 SNA sessions termination records have changed with this release. Part of the Telnet section of the TCP connection termination record has a reason code. If the connection was closed by the TN3270E server the record contains the associated reason code. See Appendix C, SMF type 119 records in *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for more details.

Note that two SMF sections appear after the host name triplet:

- Basic life of session data - transaction counts, round trip times, sum of squares
- Time bucket data

Note: Remember to configure Monitoring in the profile and create a MonitorGroup to map the group to clients using the MonitorMap statement. You do not need to set up TNSACONFIG if SNMP is not being used.

Real-time SMF information service access control

The SMF information service allows network management applications to obtain selected TCP/IP SMF records, such as SMF records supported by FTP and Telnet, in a real-time fashion. Access to this information can be controlled through an external security manager product, such as RACF, by defining the SERVAUTH profile name EZB.NETMGMT.sysname.tcpname.SYSTCPSM.

Access to these SMF records is allowed if the user ID associated with the network management application is permitted (read access) to this resource profile. In addition, to use this service, it should be enabled on the stack using the NETMONITOR SMFService statement in PROFILE.TCPIP.

Important: Existing SMF119 Telnet SNA session termination records (subtype 21) usually change with each release. Check the latest documentation for the most current SMF record formats, which can be found in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

2.3 Multiple TN3270E servers in a multiple image environment

This section provides an overview of executing multiple TN3270E servers in a sysplex environment with a server on each LPAR. This environment is illustrated in Figure 2-7.

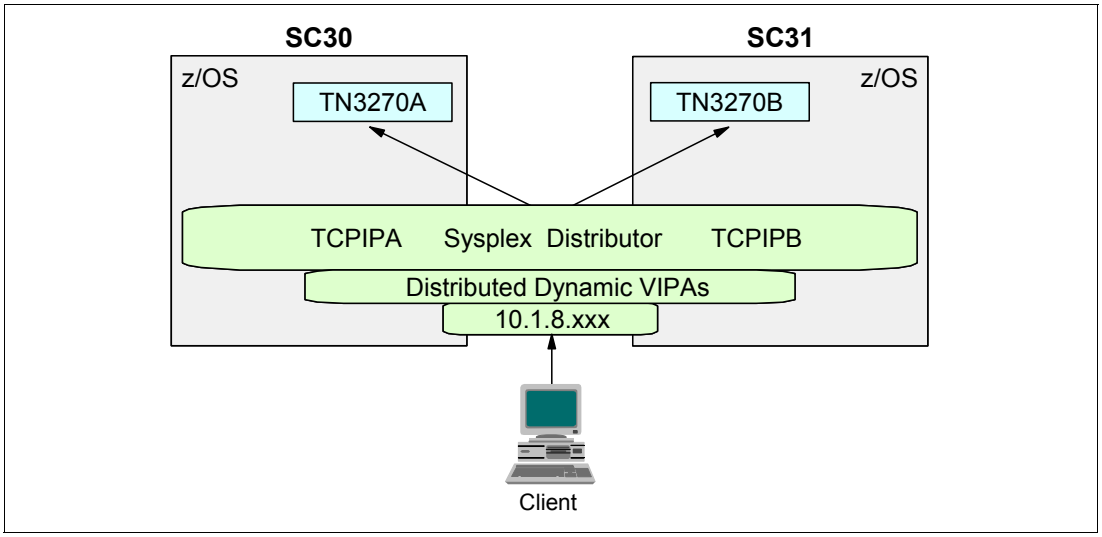


Figure 2-7 TN3270E server within the sysplex

In Figure 2-7, clients connect to the Distributed DVIPA address of the TN3270E server. Based on installation policies, the Sysplex Distributor (SD) will then direct connections to the best available TN3270E server.

Sysplex distribution takes advantage of the existence of multiple, redundant resources to provide high availability and load distribution. In our scenario all of the following are redundant resources participating in the sysplex distribution process:

- System images
- Sysplex links
- TCP/IP stacks
- Stack interfaces
- Server applications (Telnet servers, in this case)

Sysplex distribution working with Workload Manager provides intelligent, policy-based load balancing between the stacks and between the TN3270E servers.

For more information about the advantages of high availability and workload balancing, and how to implement appropriate scenarios, see the following publications:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

The following topics discuss how to implement multiple TN3270E servers in the sysplex:

- Multiple TN3270E servers within the sysplex
- Configuration of multiple TN3270E servers within the sysplex
- Activation and verification of multiple TN3270E servers in the sysplex

2.3.1 Multiple TN3270E servers within the sysplex

This scenario is based on the TN3270E server scenario that we discuss in 2.2, “TN3270E server in a single image” on page 43.

Only the additional required sysplex distribution configuration statements for the stack and for the TN3270E server are described in this section.

For this scenario we use two system images:

- SC30
- SC31

Each system uses one TCP/IP stack. The TCP/IP stack started task name is the same on both systems (*TCPIP*).

Each stack has one TN3270E server that is associated with it. The TN3270E server started task name is *TN3270A* in SC30 and *TN3270B* in SC31. Stack affinity is established for the server using the TCPIPJOBNAME statement within the TELNETGLOBALS block. We use system symbolics in the started task JCL to provide uniqueness where necessary.

Figure 2-7 on page 77 shows that on each system (SC30 and SC31) we have the following started tasks:

- TCPIPA and TCPIPB
- TN3270A and TN3270B

We designed the two stacks to back up each other. We also designed the two TN3270E servers to back up each other. Although it is not a requirement for a backup stack to distribute connections identically to the method of the primary stack, we designed our two stacks to do so. So when the primary stack fails, or otherwise relinquishes its distributor responsibilities, our backup stack will continue to distribute connections to the TN3270E servers in identical fashion as the primary.

Note: Implementing multiple, redundant TCP/IP stacks and TN3270E servers increases the effort that is required of systems personnel to maintain equivalent configurations across all participating systems. That increased effort should not be underestimated or overlooked.

Planning and design is also more complex and involves multiple departments. Mainframe systems and networking personnel must be aware of the physical network requirements. Requirements for IP subnets and IP addresses are increased by introducing sysplex distributed Dynamic VIPAs and Dynamic XCF. Operations must be made aware of changes that sysplex distribution, multiple stacks, and multiple server applications introduce to the environment. Automation and scheduling changes are also most likely to be required.

TCP/IP stack redundancy

If you have only one system image or one LPAR to consider, then multiple stacks will not improve availability. In this case a single TCP/IP stack will suffice for your environment. However, if you have two or more system images in your sysplex, we suggest that you implement a TCP/IP stack on each image and take advantage of capabilities that enable the multiple stacks to be configured to back up each other.

Stack dependencies for multiple TN3270E servers within the sysplex

Because sysplex distribution (SD) is used in this scenario, all the functionality that a TCP/IP stack needs to support SD is required, including:

- The hardware and software required for the coupling facility and XCF communication.
- XCFINIT=YES in VTAM, DYNAMICXCF in TCP/IP.
- An IP subnet and host IP addressing assigned to the XCF interfaces.
- If IBM HiperSockets™ are implemented, HiperSockets used by XCF must be consistent.
- Most of the parameters within GLOBALCONFIG, IPCONFIG, TCPCONFIG, and UDPCONFIG should be set the same on all participating stacks.

The multiple stacks must have Distributed Dynamic VIPA definitions added to support distribution to the multiple TN3270E servers.

The VIPADYNAMIC block must be coded in the backup stack in such a way that it distributes connections in a similar manner as the primary. Our scenario uses an identical process.

The introduction of sysplex distribution adds the requirement for a new IP subnet for the Distributed Dynamic VIPAs (if Dynamic VIPA has not already been implemented).

TN3270E server redundancy

For medium to large environments, it makes sense to consolidate TN3270 services into a few centralized servers—particularly to simplify server management and administration. They can be configured identically to give consistent client support and to provide redundancy for high availability. You might need or want to have more than two, depending on business and technical requirements. You might also want to implement the TN3270E server on your other mainframe systems to provide direct access to them for systems administration in cases where the centralized systems are not available.

TN3270E server dependencies for multiple servers within the sysplex

Because the two servers back up each other, certain parameters must be configured identically so that they can treat all client connections the same. The parameters that must match between the two servers include:

- In the TELNET parameter statements: Timer settings, MSG07, SNAEXT, session persistence options, session takeover options, security-related settings, Telnet device types, keyboard treatment, level of TN3270 functions supported, and other parameters that directly affect the treatment of the connections.
- In the BEGINVTAM section: All mapping statements including objects, client IDs, groups, and parameter overriding statements that affect the assignment of Logmodes, APPLs, LU names, USS tables, Interpret tables, and so on.

If these parameters differ, clients could experience differences between their sessions—and even random connection failures.

Note: Use an LU name server (LUNS) to centralize LU name allocation and avoid duplicate LU name assignments among the group of Telnet servers known as LU name requesters (LUNR). See 2.4, “Multiple TN3270E servers using LU name server and LU name requester” on page 95 for detail description about LU name server and LU name requester.

If LU name server is not used for multiple TN3270E Telnet servers, the system administrator must ensure that the LU names used are unique for each server.

Sysplex distribution

When you have multiple stacks with multiple TN3270E servers, as suggested previously, and are using IBM Parallel Sysplex® technology, we suggest that you implement sysplex distribution to provide TN3270 connection load balancing at some level: round-robin, basewlm, or serverwlm. For detailed scenarios that cover these levels of load balancing, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998.

2.3.2 Configuration of multiple TN3270E servers within the sysplex

In addition to the configuration tasks for every TN3270E server instance stated in 2.2.2, "Configuration of the TN3270E server" on page 44, the following tasks are necessary to configure multiple TN3270E servers within the sysplex:

- Customize a second TCP/IP stack started task procedure.
- Customize a second TCP/IP stack configuration profile data set.
- Customize a second TN3270E server started task procedure.
- Customize a second TN3270E server configuration profile.
- Establish an IP subnet for XCF interfaces.
- Establish an IP subnet for Dynamic VIPA.
- Enable the primary stack to support sysplex functions.
- Add a VIPADYNAMIC block to the primary TCP/IP distributing stack.
- Enable the backup stack to support sysplex functions.
- Add a VIPADYNAMIC block to the backup TCP/IP stack.
- Customize an OMPROUTE started task to support the second stack
- Customize an OMPROUTE configuration for the second OMPROUTE

Customize a second TCP/IP stack started task procedure

This second started task is for running our sysplex distribution backup stack. It can be modeled after our first (primary) started task. You can used system symbolic to provide unique names for the stacks' configuration profile data sets when define the started task procedure.

See Appendix D, "Multiple TN3270E Telnet servers and sysplex distribution using the LUNS and LUNR scenario" on page 429, for the started task procedures used for this scenario.

Customize a second TCP/IP stack configuration profile data set

This stack should be modeled after the first (primary) stack. This stack will be the backup stack. If you do not already have a second stack running, you must create it. Support for our TN3270E server application must be designed in this stack identically to that of the first stack. Obviously, there are those configuration statements that must be different between the two stacks to give them their uniqueness. Home IP addresses and possibly interface definitions are common differences. For a complete description and examples of setting up a stack, see the following publications:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

See Appendix D, "Multiple TN3270E Telnet servers and sysplex distribution using the LUNS and LUNR scenario" on page 429, for the stack profile used for this scenario.

Customize a second TN3270E server started task procedure

This second started task is for running our second TN3270E server. It can be modeled after our first server started task. You can use system symbolics to provide unique names for the stacks' configuration profile data sets when you define the started task procedure.

See Appendix D, "Multiple TN3270E Telnet servers and sysplex distribution using the LUNS and LUNR scenario" on page 429, for the started task procedures used for this scenario.

Customize a second TN3270E server configuration profile

This server should be modeled after the first server. If you do not already have a server running on the second system, you must create it. Port definitions and the mapping structure in this second server should be identical to those of the first server. Make sure that both servers treat connections coming through sysplex distribution in exactly the same way.

Note: Use an LU name server (LUNS) to centralize LU name allocation and avoid duplicate LU name assignments among the group of Telnet servers known as LU name requesters (LUNR). See 2.4, "Multiple TN3270E servers using LU name server and LU name requester" on page 95 for a detailed description about LU name server and LU name requester.

If LU name server is not used for multiple TN3270E Telnet servers, the system administrator must ensure that the LU names used are unique for each server.

In this scenario, we used different LU names for each TN3270E server. For a complete discussion and examples of setting up a TN3270E server, see 2.2.2, "Configuration of the TN3270E server" on page 44.

See Appendix D, "Multiple TN3270E Telnet servers and sysplex distribution using the LUNS and LUNR scenario" on page 429, for the basic TN3270E server profiles used for this scenario.

Establish an IP subnet for XCF interfaces

Dynamic XCF interfaces require the use of an IP subnet. Each stack is assigned a unique host address within that subnet. If you do not already have a unique IP subnet assigned for XCF, one will have to be allocated.

```
Our XCF subnet is 10.1.7.0/24
```

Establish an IP subnet for Dynamic VIPA

Dynamic VIPA and Distributed VIPA interfaces require the use of an IP subnet. If you do not already have a unique IP subnet assigned for DVIPA, one will have to be allocated:

```
Our Distributed DVIPA subnet is 10.1.8.0/24
Our Viparange   DVIPA subnet is 10.1.9.0/24
```

Enable the primary stack to support sysplex functions

Add statements to the primary stack that enable sysplex support as shown in Example 2-50.

Example 2-50 Sysplex enablement for primary stack

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN RECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.1.7.11 255.255.255.0 8
```

Add a VIPADYNAMIC block to the primary TCP/IP distributing stack

The VIPADYNAMIC statements enable the stack to distribute TN3270E server connections to the two servers. VIPADYNAMIC statements added to the primary stack are in Example 2-51.

Example 2-51 VIPADYNAMIC statements for primary stack

```
VIPADYNAMIC
  ;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.25    ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25     PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using ROUNDROBIN         -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21    ;General
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                         10.1.8.21     PORT 992 20 21 23
                  DESTIP 10.1.7.11
                         10.1.7.21
;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using BASEWLM            -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22    ;Admin
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.22     PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23  ; Payrol
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
;---------------------------------------------------------------------
  ;      Distribute to 10.1.1.10    via IP routing   (   viparoute)  -
  ;      Distribute to 10.1.1.20    via normal XCF   (no viparoute)  -
  ;---------------------------------------------------------------------
  ;VIPAROUTE      DEFINE 10.1.7.11    10.1.1.10   ; sc30's static vipa
   VIPAROUTE      DEFINE 10.1.7.21    10.1.1.20   ; sc31's static vipa
ENDVIPADYNAMIC
```

Enable the backup stack to support sysplex functions

Add statements to the backup stack that enable sysplex support. The statements that we added to the backup stack are shown in Example 2-52.

Example 2-52 Sysplex enablement for backup stack

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN RECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.1.7.21 255.255.255.0 8
```

Add a VIPADYNAMIC block to the backup TCP/IP stack

The VIPADYNAMIC statements enable the stack to take over distribution when the primary stack relinquishes that responsibility. That can happen if the stack fails or upon operator command. The statements that we added are shown in Example 2-53.

Example 2-53 VIPADYNAMIC statements for backup stack

```
VIPADYNAMIC
;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPABACKUP     MOVE IMMED 255.255.255.0 10.1.8.25    ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25     PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPABACKUP     MOVE IMMED 255.255.255.0 10.1.8.21    ;General
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                         10.1.8.21     PORT 992 20 21 23
                  DESTIP 10.1.7.11
                         10.1.7.21
;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPABACKUP     MOVE IMMED 255.255.255.0 10.1.8.22    ;Admin
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.22     PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPABACKUP     MOVE IMMED 255.255.255.0 10.1.8.23  ; Payrol
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
  ;-------------------------------------------------------------------
  ;      Distribute to 10.1.1.10    via IP routing   (   viparoute)  -
  ;      Distribute to 10.1.1.20    via normal XCF   (no viparoute)  -
  ;-------------------------------------------------------------------
   VIPAROUTE      DEFINE 10.1.7.11    10.1.1.10   ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.1.7.21    10.1.1.20   ; sc31's static vipa
ENDVIPADYNAMIC
```

Customize an OMPROUTE started task to support the second stack

This second started task is for running our second OMPROUTE. It can be modeled after our first started task. You can use system symbolic to provide unique names for the OMPROUTE configuration data sets when define the started task JCL.

See Appendix D, “Multiple TN3270E Telnet servers and sysplex distribution using the LUNS and LUNR scenario” on page 429, for the started task procedures used for this scenario.

Customize an OMPROUTE configuration for the second OMPROUTE

This second configuration data set can be modeled after the first. Obviously, there are statements that must be different between the two configurations to give them their uniqueness: interface IP addresses and router IDs are examples. For details about configuring OMPROUTE, see the following publications:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Additional statements must be added to support the Dynamic XCF interfaces and the Dynamic VIPA interfaces. Example 2-54 shows the additional OMPROUTE statements for the primary stack.

Example 2-54 OMPROUTE additional statements to support D-XCF and DVIPA: for primary stack

```
;
Global_Options Ignore_Undefined_Interfaces=yes;
;
; ************************************************************
; *Dynamic VIPA requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.8.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.8.* subnet should be dedicated to D-VIPA use.
; ************************************************************
; Dynamic vipa VIPADEFINE
ospf_interface ip_address=10.1.8.*
           subnet_mask=255.255.255.0
           Advertise_VIPA_Routes=HOST_ONLY
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=65535;
;
; ************************************************************
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
```

```
; *Another way to accomplish all this is to just code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
interface  ip_address=10.1.7.*
           subnet_mask=255.255.255.0
           mtu=65535;
```

Example 2-55 shows the additional OMPROUTE statements for the backup stack.

Example 2-55 OMPROUTE additional statements to support D-XCF and DVIPA: for backup stack

```
;
Global_Options Ignore_Undefined_Interfaces=yes;
;
; ************************************************************
; *Dynamic VIPA Range requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.8.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.8.* subnet should be dedicated to D-VIPA use.
; ************************************************************
; Dynamic vipa VIPADEFINE
ospf_interface ip_address=10.1.8.*
           subnet_mask=255.255.255.0
           Advertise_VIPA_Routes=HOST_ONLY
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=65535;
; ************************************************************
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to just code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
interface  ip_address=10.1.7.*
           subnet_mask=255.255.255.0
           mtu=65535;
```

See Appendix D, “Multiple TN3270E Telnet servers and sysplex distribution using the LUNS and LUNR scenario” on page 429 for the basic OMPROUTE server profiles used for this scenario.

2.3.3 Activation and verification of multiple TN3270E servers in the sysplex

In this section, we show how to activate, verify, and manage multiple TN3270E servers within the sysplex. A number of NETSTAT displays can be used to show the status of Dynamic and Distributed VIPA connections. The format of the system NETSTAT command is:

```
D TCPIP,procname,N,command,option
```

For complete details about the NETSTAT command, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

All of the verification tasks for a stand-alone TN3270E server apply. See 2.2.4, "Verification of the TN3270E server" on page 53.

Then perform these additional tasks to verify multiple TN3270E servers within the sysplex:

- Start the second (backup) stack on the second system.
- Start the second TN3270E server on the second system.
- Use TELNET CONN displays to show TN3270 connections.
- Use NETSTAT VCRT to show dynamic VIPA connection routing table.
- Use NETSTAT VDPT to show dynamic VIPA destination port table.
- Use NETSTAT VIPADCFG to show current dynamic VIPA configuration.
- Use NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE.

Start the second (backup) stack on the second system

Issue the MVS START command on the second system for the second TCPIP and OMPROUTE:

```
S TCPIPA
S OMPA
```

Then look for the expected initialization messages.

Start the second TN3270E server on the second system

Issue the MVS START command on the second system for the second TN3270E server:

```
S TN3270A,PROFILE=TELNA30B
```

Then look for the expected initialization messages.

Use TELNET CONN displays to show TN3270 connections

In preparation for the following displays, we connected two clients to the Dynamic VIPA 10.1.8.21 that are distributed by SC30, which is running with stack profile PROFA30 and TN3270A profile TELNA30B. The DVIPA 10.1.8.21 represents TSO on SC30 (SC30TS). The distribution method for 10.1.8.21 port 992 is set to ROUNDROBIN for our test to show the result of distribution upon our two client connections. Using the TSO Telnet client on SC31, we connected to 10.1.8.21 port 922 (expecting to be mapped to TSO on SC30). Using the TSO Telnet client on SC32, we again connected to 10.1.8.21 port 992 (mapping to TSO on SC30).

Our LU naming convention includes the system name as part of the LU name. This helps us determine to which system the connection is made.

Example 2-56 shows a display of the connections on SC30, and Example 2-58 shows CONN=E2 from SC31 (10.1.1.20) mapped to TSO on SC30, with an LU name of SC30B*S*02. The S indicates the LU pool for secure port 992 as expected.

A display of the connections on SC31 in Example 2-57 and Example 2-59 on page 88 show CONN=6E from SC32 (10.1.2.30) mapped to TSO on SC30, with an LU name of SC31BS03, the *S* indicating the LU pool for secure port 992 as expected. The connection summary report for SC30 is shown in Example 2-56.

Example 2-56 SC30 connection to Dynamic VIPA 10.1.8.21, summary

```
D TCPIP,TN3270A,CONN
EZZ6064I TN3270A CONN DISPLAY 408
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000OE2    ::FFFF:10.1.1.20..1031
                                  SC30BS02 SC30TS03  TA3 SNX32702
----- PORT:   992  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
9 OF 9 RECORDS DISPLAYED
```

Example 2-57 shows the connection summary report for SC31.

Example 2-57 SC31 connection to Dynamic VIPA 10.1.8.21, summary

```
RO SC31,D TCPIP,TN3270B,CONN
   D TCPIP,TN3270B,CONN
   EZZ6064I TN3270B CONN DISPLAY 994
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- --------------------- -------- --------  --- --------
   0000006E    ::FFFF:10.1.2.30..1039
                                     SC31BS03 SC30TS02  TA3 SNX32702
   ----- PORT:   992  ACTIVE           PROF: CURR CONNS:      1
   ------------------------------------------------------------
   9 OF 9 RECORDS DISPLAYED
```

Example 2-58 shows detailed information about one single connection on SC30.

Example 2-58 SC30 connection to Dynamic VIPA with detailed information

```
RO SC30,D TCPIP,TN3270A,CONN,CONN=E2,DET
   D TCPIP,TN3270A,CONN,CONN=E2,DET
   EZZ6065I TN3270A CONN DISPLAY 433
     CONNECTED: 12:00:45 09/29/2010 STATUS:  SESSION ACTIVE
     CLIENT IDENTIFIER FOR CONN: 00000OE2   SECLABEL: **N/A**
       CLIENTAUTH USERID: **N/A**
       HOSTNAME: NO HOSTNAME
       CLNTIP..PORT: ::FFFF:10.1.1.20..1031
       DESTIP..PORT: ::FFFF:10.1.8.21..992
       LINKNAME: VIPL0A010815
     PORT:   992 QUAL: NONE
       AFFINITY: TCPIPA
       STATUS: ACTIVE  SECURE
       ACCESS: NON-SECURE
     PROTOCOL: TN3270            DEVICETYPE: IBM-3278-2-E
```

```
    TYPE: TERMINAL GENERIC
    OPTIONS: -TET----   327OE FUNCTIONS: *N/A*
                       NEWENV FUNCTIONS: --
  LUNAME: SC30BS02
  APPL: SC30TS03
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
   LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
 MAPPING TYPE:  CONN IDENTIFIER
                     OBJECT   ITEM SPECIFIC     OPTIONS
   LUMAP GEN:  NL (NULL)
                    >*DEFLUS*                   --------
   DEFLT APPL: DG GENERALUSER
                     SC30TS                     --------
   USS TABLE:  **N/A**
   INT TABLE:  **N/A**
   MONGROUP:   DG GENERALUSER
                     SNAANDIP                   --------
     PERIOD:      120 MULT:        5
           S/W AVG LOC AVG  SUM R/T      SSQ R/T  ST DEV
           ======= ======= ======== ============ =======
     SNA:        0       0        0            0       0
     IP:         0       0        0            0       0
     TOTAL:      0       0        0            0       0
     COUNT:      0       0
     BUCKET1    BUCKET2    BUCKET3    BUCKET4    BUCKET5
            1          2          3          4     NO LMT
            0          0          0          0          0
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  SSS------  *---ST-  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  SSS*D****  *P**STS  SDD* TP-CURR
    PARMSGROUP: DG GENERALUSER
  *------  ------------  ---  -B-------  -------  ---- NOSSL
  *M*****  **TSBTQ***RT  ECF  SBS*D****  *P**STS  SDD* <-FINAL
 51 OF 51 RECORDS DISPLAYED
```

Example 2-59 shows detailed information about a single connection on SC31.

Example 2-59 SC31 connection to Dynamic VIPA with detailed information

```
RO SC31,D TCPIP,TN327OB,CONN,CONN=6E,DET
  D TCPIP,TN327OB,CONN,CONN=6E,DET
  EZZ6065I TN327OB CONN DISPLAY 007
    CONNECTED: 13:07:19  09/29/2010  STATUS: SESSION ACTIVE
    CLIENT IDENTIFIER FOR CONN: 0000006E   SECLABEL: **N/A**
      CLIENTAUTH USERID: **N/A**
      HOSTNAME: NO HOSTNAME
      CLNTIP..PORT: ::FFFF:10.1.2.30..1039
      DESTIP..PORT: ::FFFF:10.1.8.21..992
      LINKNAME: VIPL0A010815
    PORT:   992 QUAL: NONE
      AFFINITY: TCPIPB
```

```
   STATUS: ACTIVE  SECURE
   ACCESS: NON-SECURE
 PROTOCOL: TN3270            DEVICETYPE: IBM-3278-2-E
   TYPE: TERMINAL GENERIC
   OPTIONS: -TET----   3270E FUNCTIONS: *N/A*
                      NEWENV FUNCTIONS: --
 LUNAME: SC31BS03
 APPL: SC30TS02
  USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
  LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
MAPPING TYPE:  CONN IDENTIFIER
                    OBJECT   ITEM SPECIFIC     OPTIONS
  LUMAP GEN:  NL (NULL)
                   >*DEFLUS*                   --------
  DEFLT APPL: DG GENERALUSER
                    SC30TS                     --------
  USS TABLE:  **N/A**
  INT TABLE:  **N/A**
  MONGROUP:   DG GENERALUSER
                    SNAANDIP                   --------
    PERIOD:      120 MULT:        5
          S/W AVG LOC AVG  SUM R/T      SSQ R/T  ST DEV
          ======= ======= ======== ============ =======
    SNA:        0       0        0            0       0
    IP:         0       0        0            0       0
    TOTAL:      0       0        0            0       0
    COUNT:      0       0
     BUCKET1    BUCKET2    BUCKET3    BUCKET4    BUCKET5
           1          2          3          4     NO LMT
           0          0          0          0          0
   PARMS:
 PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
(LMTGCAK)(OATSKTQSSHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
 -------  ------------  ---  ---------  -------  ----
 *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
 -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
 -M-----  ---S--------  --F  SSS------  *---ST-  S--- *TPARMS
 *M*****  **TSBTQ***RT  ECF  SSS*D****  *P**STS  SDD* TP-CURR
   PARMSGROUP: DG GENERALUSER
 *------  ------------  ---  -B-------  -------  ---- NOSSL
 *M*****  **TSBTQ***RT  ECF  SBS*D****  *P**STS  SDD* <-FINAL
51 OF 51 RECORDS DISPLAYED
```

Use NETSTAT VCRT to show dynamic VIPA connection routing table

VCRT displays the dynamic VIPA connection routing table information.

For each table entry that represents an established dynamic VIPA connection or an affinity created by the passive-mode FTP, the DETAIL suboption additionally displays the policy rule, action information, and routing information. For each entry that represents an affinity created by the TIMEDAFFINITY parameter on the VIPADISTRIBUTE profile statement, it displays the preceding information plus the affinity-related information.

Example 2-60 shows the connections at the time the VCRT command was issued. Notice that the distributing stack knows about all of the connections because it is managing them.

Example 2-60 NETSTAT VCRT on system SC30

```
RO SC30,D TCPIP,TCPIPA,N,VCRT
   D TCPIP,TCPIPA,N,VCRT
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.1.8.21..992
     SOURCE:  10.1.1.20..1031
     DESTXCF: 10.1.7.11
   DEST:      10.1.8.21..992
     SOURCE:  10.1.2.30..1039
     DESTXCF: 10.1.7.21
   2 OF 2 RECORDS DISPLAYED
   END OF THE REPORT
```

Notice that the non-distributing stack shows only the connections that have been distributed to it, as shown in Example 2-61.

Example 2-61 NETSTAT VCRT on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VCRT
   D TCPIP,TCPIPB,N,VCRT
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.1.8.21..992
     SOURCE:  10.1.2.30..1039
     DESTXCF: 10.1.7.21
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

Use NETSTAT VDPT to show dynamic VIPA destination port table

VDPT displays the dynamic VIPA destination port table information.

If the DETAIL suboption is specified, the output will contain policy action information, target responsiveness values, and a WQ value (on a separate line). If DETAIL is not specified, the output will not contain policy action information or target responsiveness and WQ values.

Example 2-62 shows the port table entries at the time of issuing the VDPT command. SC30 is currently the distributor, so it shows the ports being distributed and whether there is a ready listener on the port.

Note: The TOTALCONN field indicates the total number of connections there have been since the distribution started for the port. It does *not* represent the *current* number of connections.

Example 2-62 NETSTAT VDPT on system SC30

```
RO SC30,D TCPIP,TCPIPA,N,VDPT
   D TCPIP,TCPIPA,N,VDPT
   DYNAMIC VIPA DESTINATION PORT TABLE:
   DEST:        10.1.8.21..20
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
```

```
  DEST:        10.1.8.21..20
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
    FLG: ROUNDROBIN
  DEST:        10.1.8.21..21
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
    FLG: ROUNDROBIN
  DEST:        10.1.8.21..21
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 001  WLM: 01  TSR: 100
    FLG: ROUNDROBIN
  DEST:        10.1.8.21..23
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000000  RDY: 001  WLM: 01  TSR: 100
    FLG: ROUNDROBIN
  DEST:        10.1.8.21..23
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 001  WLM: 01  TSR: 100
    FLG: ROUNDROBIN
  DEST:        10.1.8.21..992
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000006  RDY: 001  WLM: 01  TSR: 100
    FLG: ROUNDROBIN
  DEST:        10.1.8.21..992
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000005  RDY: 001  WLM: 01  TSR: 100
    FLG: ROUNDROBIN
  ...
```

SC31 is not a distributor at the moment, therefore it shows no information, as shown in Example 2-63.

Example 2-63 NETSTAT VDPT on system SC31

```
D TCPIP,TCPIPB,N,VDPT
DYNAMIC VIPA DESTINATION PORT TABLE:
0 OF 0 RECORDS DISPLAYED
END OF THE REPORT
```

Use NETSTAT VIPADCFG to show current dynamic VIPA configuration

VIPADCFG displays the current dynamic VIPA configuration information from the perspective of the stack on which the command is entered.

Examples of VIPADCFG showing configuration information follow. The primary distributor shows VIPA DEFINE, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 2-64.

Example 2-64 NETSTAT VIPADCFG on system SC30

```
RO SC30,D TCPIP,TCPIPA,N,VIPADCFG
D TCPIP,TCPIPA,N,VIPADCFG
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IPADDR/PREFIXLEN: 10.1.8.24/24
      RANK: 200  MOVEABLE: IMMEDIATE  SRVMGR: NO
```

```
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.21/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
    IPADDR/PREFIXLEN: 10.1.8.22/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
    IPADDR/PREFIXLEN: 10.1.8.23/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
    IPADDR/PREFIXLEN: 10.1.8.25/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
  VIPA RANGE:
    IPADDR/PREFIXLEN: 10.1.9.0/24
      MOVEABLE: NONDISR
 VIPA DISTRIBUTE:
...
   DEST:        10.1.8.21..23
     DESTXCF:   10.1.7.11
       SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
   DEST:        10.1.8.21..23
     DESTXCF:   10.1.7.21
       SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
  DEST:        10.1.8.21..992
    DESTXCF:   10.1.7.11
      SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
  DEST:        10.1.8.21..992
    DESTXCF:   10.1.7.21
      SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
...
  VIPA ROUTE:
      DESTXCF:   10.1.7.21
      TARGETIP:  10.1.1.20
END OF THE REPORT
```

The backup stack shows VIPA BACKUP, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 2-65.

Example 2-65 NETSTAT VIPADCFG on system SC31

```
D TCPIP,TCPIPB,N,VIPADCFG
   DYNAMIC VIPA INFORMATION:
     VIPA BACKUP:
       IPADDR/PREFIXLEN: 10.1.8.21
         RANK: 001  MOVEABLE:            SRVMGR:
       IPADDR/PREFIXLEN: 10.1.8.22
         RANK: 001  MOVEABLE:            SRVMGR:
       IPADDR/PREFIXLEN: 10.1.8.23
         RANK: 001  MOVEABLE:            SRVMGR:
       IPADDR/PREFIXLEN: 10.1.8.25
         RANK: 001  MOVEABLE:            SRVMGR:
     VIPA RANGE:
       IPADDR/PREFIXLEN: 10.1.9.0/24
         MOVEABLE: NONDISR
     VIPA DISTRIBUTE:
    ...
        DEST:        10.1.8.21..23
          DESTXCF:   10.1.7.11
            SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
        DEST:        10.1.8.21..23
          DESTXCF:   10.1.7.21
```

```
          SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.1.8.21..992
        DESTXCF:   10.1.7.11
          SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.1.8.21..992
        DESTXCF:   10.1.7.21
          SYSPT:   YES  TIMAFF: NO    FLG: ROUNDROBIN
...
  VIPA ROUTE:
    DESTXCF:     10.1.7.11
      TARGETIP:  10.1.1.10
END OF THE REPORT
```

Use NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE

VIPADYN displays the current dynamic VIPA and VIPAROUTE information from the perspective of the stack on which the command is entered. There are two suboptions available to filter the output:

- DVIPA: Displays the current dynamic VIPA information only
- VIPAROUTE: Displays the current VIPAROUTE information only

Example 2-66 shows SC30, NETSTAT VIPADYN with FTP and TN3270 DVIPA addresses.

Example 2-66 NETSTAT VIPADYN on system SC30

```
D TCPIP,TCPIPA,N,VIPADYN
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
    IPADDR/PREFIXLEN: 10.1.8.24/24
      STATUS:  ACTIVE     ORIGIN: VIPABACKUP       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
  VIPA ROUTE:
    DESTXCF: 10.1.7.21
      TARGETIP: 10.1.1.20
      RTSTATUS: ACTIVE
  6 OF 6 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-67 shows SC30, NETSTAT VIPADYN,DVIPA, with filters on DVIPA only.

Example 2-67 NETSTAT VIPADYN,DVIPA on system SC30

```
D TCPIP,TCPIPA,N,VIPADYN,DVIPA
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
```

```
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
    IPADDR/PREFIXLEN: 10.1.8.24/24
      STATUS:  ACTIVE     ORIGIN: VIPABACKUP       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 11/20/2008 16:47:03
  5 OF 5 RECORDS DISPLAYED
```

Example 2-68 shows SC30, NETSTAT VIPADYN,VIPAROUTE with filters on VIPA ROUTE only.

Example 2-68 NETSTAT VIPADYN,VIPAROUTE on system SC30

```
D TCPIP,TCPIPA,N,VIPADYN,VIPAROUTE
  VIPA ROUTE:
    DESTXCF: 10.1.7.21
      TARGETIP: 10.1.1.20
      RTSTATUS: ACTIVE
  1 OF 1 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-69 shows SC31, NETSTAT VIPADYN with ftp and TN3270 DVIPA addresses.

Example 2-69 NETSTAT VIPADYN on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN
  D TCPIP,TCPIPB,N,VIPADYN
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.24/24
      STATUS:  ACTIVE     ORIGIN:                  DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
  VIPA ROUTE:
    DESTXCF: 10.1.7.11
      TARGETIP: 10.1.1.10
      RTSTATUS: ACTIVE
  6 OF 6 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-70 shows SC31, NETSTAT VIPADYN,DVIPA with filters on DVIPA only.

Example 2-70 NETSTAT VIPADYN,DVIPA on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
  D TCPIP,TCPIPB,N,VIPADYN,DVIPA
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.24/24
      STATUS:  ACTIVE     ORIGIN:                  DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
      ACTTIME: 11/20/2008 17:09:15
  5 OF 5 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-71 shows SC31, NETSTAT VIPADYN,VIPAROUTE with filters on viparoute only.

Example 2-71 NETSTAT VIPADYN,VIPAROUTE on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
  D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
  VIPA ROUTE:
    DESTXCF: 10.1.7.11
      TARGETIP: 10.1.1.10
      RTSTATUS: ACTIVE
  1 OF 1 RECORDS DISPLAYED
  END OF THE REPORT
```

2.4 Multiple TN3270E servers using LU name server and LU name requester

This section provides an overview of executing multiple TN3270E servers using TN3270 LU name server (LUNS) and TN3270 LU name requester (LUNR). It includes the following topics:

- Description of TN3270E servers using LU name server and requester
- Configuration of TN3270E servers within sysplex using LU name server and requester
- Activation and verification of LU name server and requester within sysplex

For more detailed information about the description and configuration of LU name server and LU name requester, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

2.4.1 Description of TN3270E servers using LU name server and requester

An LU is the endpoint of an SNA session. The LU name in a SNA network must be unique. As discussed in 2.3.2, “Configuration of multiple TN3270E servers within the sysplex” on page 80, you receive more than one alert to not use the same LU names in multiple TN3270E servers. Defining multiple TN3270E servers and maintaining the uniqueness of the LU names in multiple TN3270E servers environment requires careful planning regarding how LU names are assigned by the individual TN3270 servers in the sysplex.

TN3270 LU name server

A TN3270 server manages the allocation of LU names in multiple TN3270E servers in a sysplex. The Telnet server used for centralized LU allocation is referred to as an LU name server (LUNS).

The LU name server accepts request to allocate an LU name from the a TN3270E server that accepts connection requests from TN3270E clients. The TN3270E workstation server is referred to as LU name requester (LUNR). Instead of directly assigning an LU name, the request to an LU name server ensures no duplicate LU name assignments, as shown in Figure 2-8.

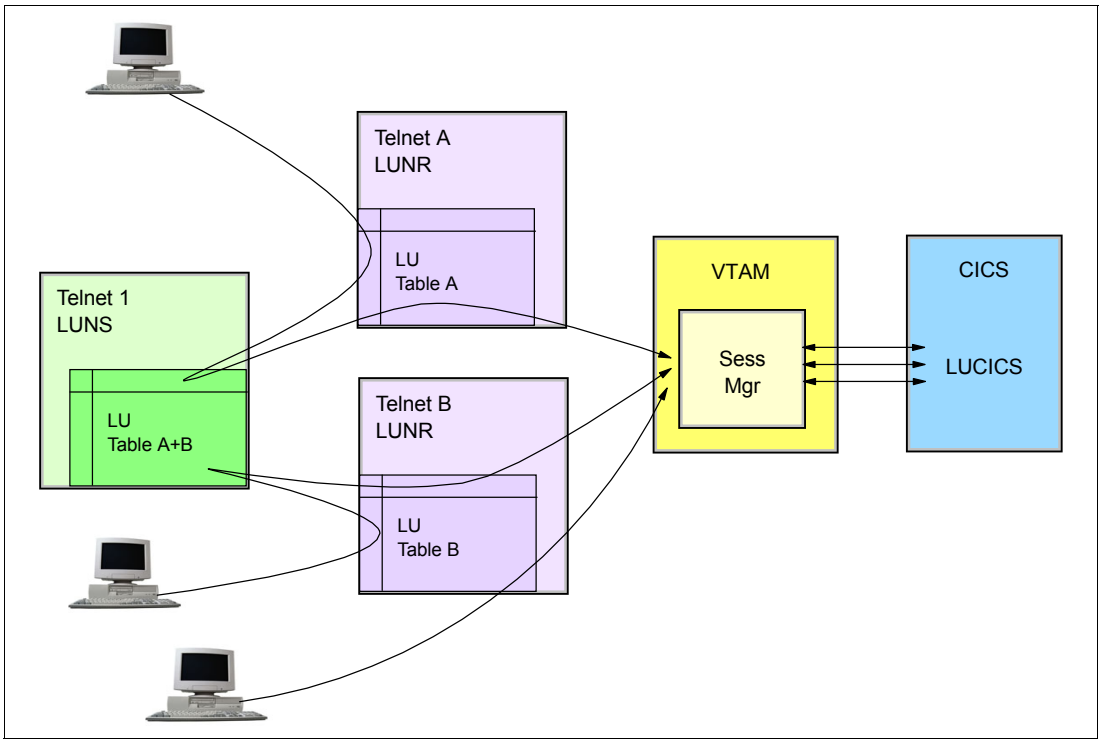


Figure 2-8 TN3270E LU name server implementation

In most cases, all LU groups are defined at the LU name requester. Existing LU group definitions are considered local groups. A new set of LU group definitions define the shared LU groups. The shared LU group definitions are sent to the LU name server. It is possible to define a mixture of shared and local LU groups. Only the connections that are mapped to shared LU groups use the LU name server services for LU assignment. If a connection maps to a local LU group, the Telnet server assigns the LU name directly from its pool, as it always has.

Shared LU groups

New LU group object are used at the LU name requester to indicate that the set of LU names in that group need to be coordinated by the LU name server. These object types are called *shared* LU groups. Shared LU group definitions can be the same on multiple Telnet servers. The shared LU group definitions are sent to the LU name server for central management of LU names. If there is no active LU name server, the LU name requester waits and the profile remains in PENDING state until an LU name server becomes active. Example 2-72 shows the six statements that are used for defining shared LUs.

Example 2-72 Shared LU definition statements

```
SLUGROUP                    ...ENDSLUGROUP
SPRTGROUP                   ...ENDSPRTGROUP
SDEFAULTLUS                 ...ENDSDEFAULTLUS
SDEFAULTPRT                 ...ENDSDEFAULTPRT
SDEFAULTLUSSPEC             ...ENDSDEAFULTSSPEC
SDEFAULTPRTSPEC             ...EDNSDEFAULTPRTSPEC
```

Note: Shared LUs are defined only in the LU name requester.

Non-shared LU groups

Existing non-shared LU groups continue to be supported. Assignment of LU names from non-shared LU groups is managed locally on the Telnet server where the LU group is defined. Other Telnet servers in the sysplex remain unaware of non-shared LU name assignments. LU names in non-shared LU groups still must be administered manually to prevent duplicate client assignment. LU names should not be defined in both shared and non-shared LU name groups. Otherwise, duplicate LU assignments are still possible between non-shared and shared LU name groups.

Note: Non-shared LUs can be defined either on the TN3270 LU name requester or the LU name server or both.

LU name server and LU name requester interaction

The Telnet LU name server keeps track of LU group definitions by LU name requester. The LU name must be defined by an LU name requester for the LU name server to allocate the LU name to that LU name requester. The LU name requester request contains the LU group name that should be searched at the LU name server. If multiple LUNRs define the same LU name or the same LU name is defined in several LU groups, the LU name server ensures that only one LU name requester is using the LU name.

Figure 2-9 on page 98 shows a client connection that is accepted at TelnetA and maps to SLUGRP1. The LU name request to the LU name server includes the LU name requester name, TelnetA, and the LU group name SLUGRP1. The LU name server searches only TelnetA-SLUGRP1 for a possible LU allocation and makes sure that LU is not allocated to any other connection across all LUNRs. Assume LU1 is allocated and another client connection is accepted at TelnetB and maps to SLUGRP2. The LU name request to the LU name server includes TelnetB and SLUGRP2 and searches only TelnetB-SLUGRP2 for an available LU name. LU1 is already in use, and LU5 is available. The LU name server allocates LU5 for TelnetB to use.

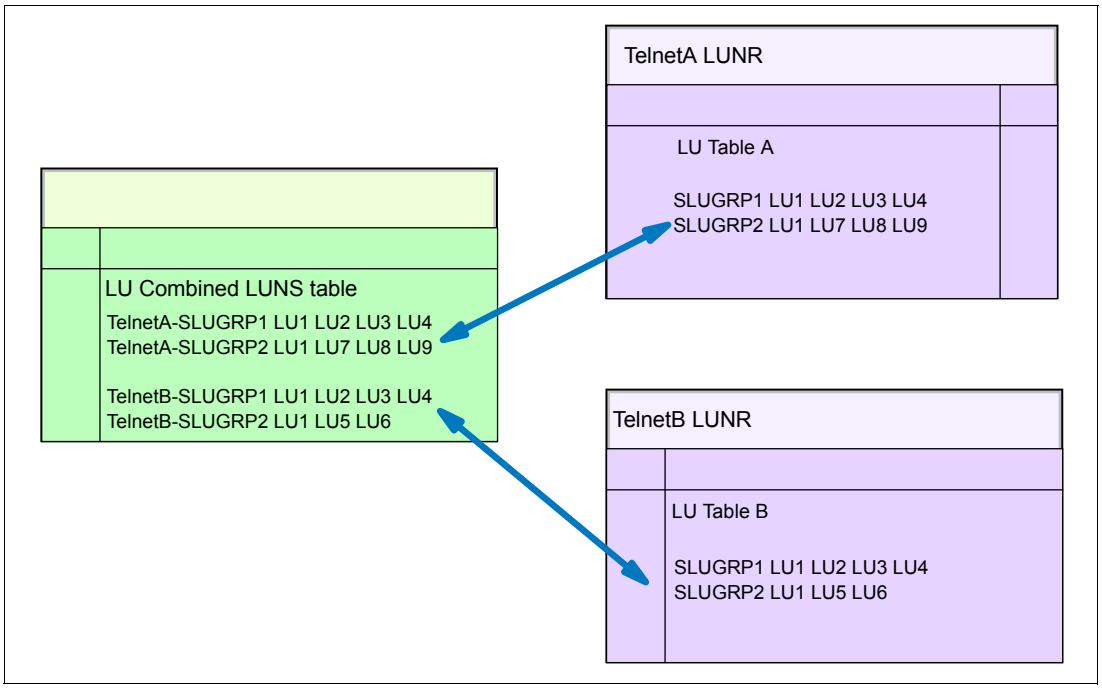


Figure 2-9 LU name server management of LU group

The shared LUs are defined at the LU name requester, and their definition is sent to the LU name server over an IP connection. The LU name requester must know the IP address of the LU name server and the of port the LU name server administrative listener. The LU name requester owns all the LU definition and sends the LU group definitions to the LU name server. Depending on the number of defined LUs during system startup, considerable volume of data can flow between the LU name server and the LU name requester.

LU name server and LU name requester design considerations

TN3270E Telnet servers can play different roles in a sysplex. If no changes are made to a Telnet server's configuration, it will not join an XCF group and will not participate in coordinated LU name assignment. These servers are now called *Classic* Telnet servers.

A TN3270E Telnet server that joins an XCF group cannot display the status of all of the members in that group and is called an *XCF Telnet* server. An XCF Telnet is capable of defining shared LU groups for use on a Telnet port.

A TN3270E Telnet server that joins an XCF group and is configured to coordinate LU name assignments is called an *XCF LU name server Telnet server*.

Any given Telnet server can be configured to participate in any combination of these roles. An LU name requester Telnet can have ports that use only shared LU name groups or a mixture

of shared and non-shared LU name groups. An LU name server Telnet can also be an LU name requester Telnet, or it might be a dedicated LU name server Telnet.

Running Telnet as only an LU name server in its own address space has the following advantages:

- Telnet port server functions do not compete with the LU name server for resources within the address space.
- You can separate Telnet roles, which makes problem diagnosis easier.
- You can stop and restart the Telnet port servers without stopping the Telnet LU name server.
- You can set the Telnet LU name server priority to a different priority than that of Telnet port servers.

A sysplex can have only one active LU name server, but to avoid single point failure, you need to define one or more backup LU name server in the sysplex. You can configure an LU name server as a primary or a backup LU name server. When the primary LU name server starts, it determines whether there is already an active LU name server in the XCF group. If there is not, that primary LU name server attempts to become the active LU name server. If there is already an active LU name server, the primary LU name server becomes a standby LU name server. If several Telnet servers that are designated as a primary LU name server are started concurrently, then the first server to connect becomes the active LU name server and the others become standby servers.

When a backup LU name server starts, if there is no active LU name server in the XCF group, the backup LU name server have a JOINED state and wait for the active LU name server. After the primary LU name server starts and becomes the active LU name server, the backup LU name server can change the state from JOINED to STANDBY and act as standby LU name server. When the active LU name server fails, the standby LU name server becomes the active LU name server.

When you define the LU name server and LU name requester, consider the following notes:

- A TN3270E can be defined as a dedicated LU name server.
- A TN3270E can act simultaneously as an LU name server and LU name requester.
- You can define non-shared LUs in an LU name server.
- An LU name requester can support both shared and unshared LU groups.
- You can only define shared LUs in an LU name requester.

LU name server and LU name requester configuration architecture

Figure 2-10 shows the roles for a TN3270E server in a sysplex.

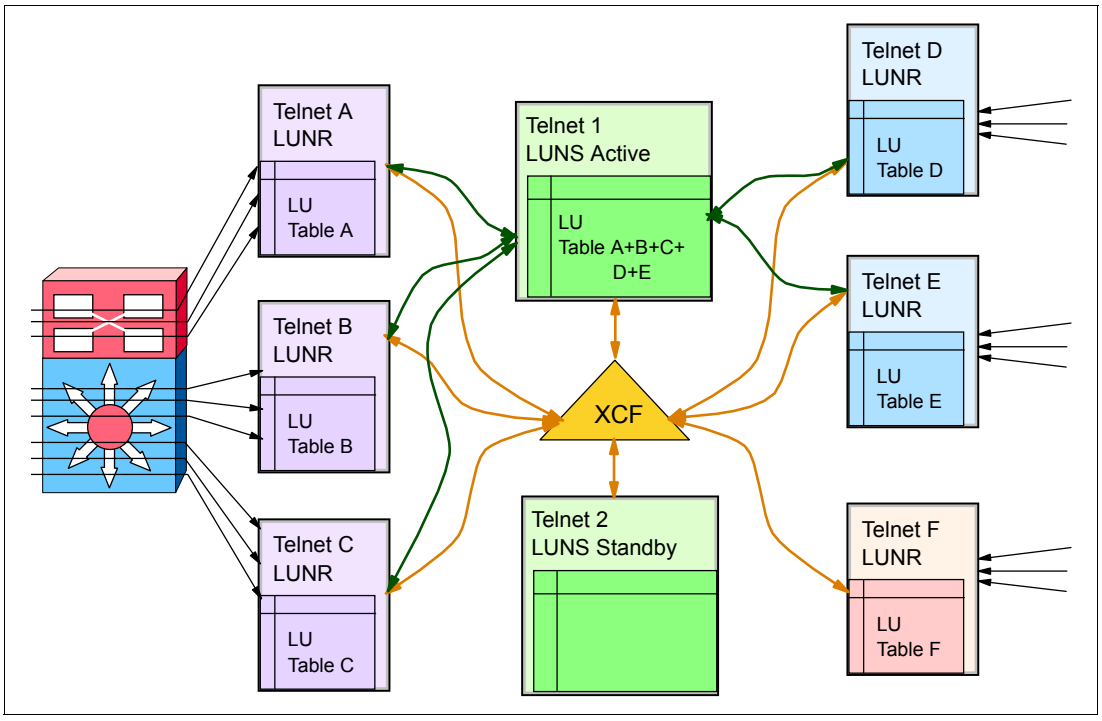


Figure 2-10 LU name server and LU name requester configuration

The green (Telnet 1 and Telnet 2) are the primary and standby LU name server. Telnet 1 is the *active* LU name server in the group and Telnet 2 is the *standby* LU name server. Telnet 2 takes over automatically as the active LU name server in the group if the current active LU name server in fails.

The purple (Telnet A, Telnet B, and Telnet C) servers are all members of a Telnet XCF group. The three Telnet servers (A, B, and C) are all serving the same port on the same IP address with identical LU group definitions and mapping statements.

The blue (Telnet D and Telnet E) LUNRs are each serving different Telnet ports with different shared LU group definitions and mapping statements.

The red (Telnet F) has not joined the group and is serving another Telnet port with its own LU group definitions and mapping statements and has no communication with the group.

The LU name server ensure that LU names defined in shared groups are assigned to only one LU name requester at a time, regardless of whether the Telnet F server has joined a group.

Every TN3270E server has two connections:

- The classic connection to the XCF
- The connection to the administrative port of the LU name server

When the standby LU name server takes the role of the primary LU name server, the administrative connection establishes a socket connection with the standby LU name server.

Telnet XCF group

The LU name server and LU name requester use only the XCF group services and requires only XCF-Local mode. This enables a wide range of Telnet group sizes, from two Telnet servers on a single stand-alone machine, to many Telnet servers on each machine in a large sysplex. Each member of the group receives notification from XCF when another member joins or leaves the group and when any member updates its member user status field. Telnet uses the XCF member status field to communicate the roles each member is playing in the group, the state of each member's LU name requester and LU name server, the instance count of the LU name server each member currently recognizes as the controlling LU name server in the group, and various status and problem flags to which other members might need to respond.

Note: A coupling facility is not needed to implement a TN3270E LU name server and LU name requester

Several independent Telnet XCF groups can be created concurrently by defining multiple Telnet XCF group names, or subplexes, in the sysplex. You might want to create multiple XCF groups to isolate internal client and external client Telnet XCF groups. All the LU name requester members of a group must be able to create TCP connections to the LU name server. All the members of a group should be in the same VTAM NETID.

XCFGROUP statement

The XCFGROUP statement determines whether the TN3270E server joins the XCF group and the roles it carries out (LU name server, LU name requester, or both).

The XCFGROUP statement is coded in TELNETGLOBALS statement block and, as shown in Figure 2-11 on page 102, is divided into the following parts:

- The XCFGROUP

  The XCFGROUP is the common part of the XCFGROUP statement and is coded for either the LU name server or the LU name requester.

- The LU name server (LUNS)

  Coded when you want to define an LU name server.

- The LU name requester (LUNR)

  Specifies the TN3270E as an LU name requester.

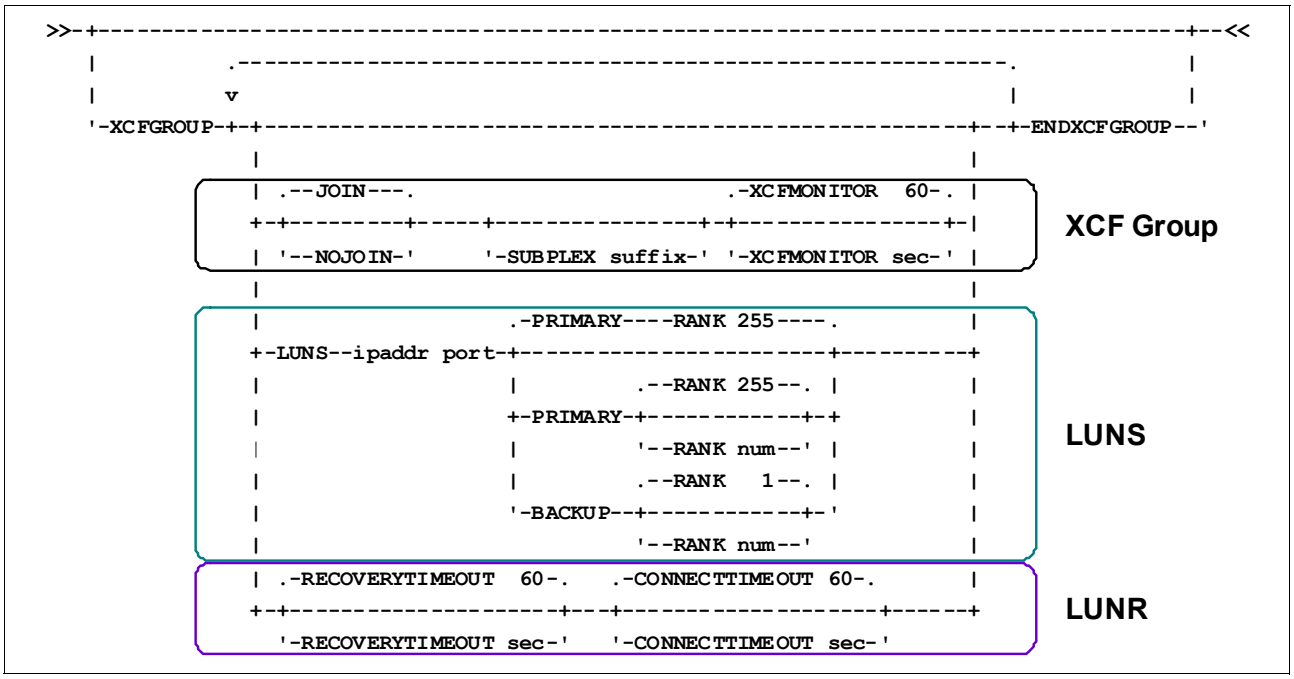


Figure 2-11 XCFGROUP statement for LU name server and LU name requester

The TN3270E server must join the XCF group and specify LU name server-specific parameters to be an LU name server. These parameters are the IPADDR and port specifying where this Telnet listens for LU name requester connection requests when it becomes the active LU name server.

PRIMARY specifies that this Telnet becomes the active LU name server at job initiation if there is not already an active LU name server. If you specify more than one TN3270E server as primary, both TN3270E servers enter to LU name server START state with an LU name server count of 1 and use that count to compete for the sysplex enqueue. The TN3270E that wins the race moves to *ACTIVE* with an LU name server count of 1. The second LU name server loses and moves to *STANDBY*.

BACKUP specifies that this Telnet moves to LU name server *STANDBY* state at job initiation.

2.4.2 Configuration of TN3270E servers within sysplex using LU name server and requester

In this section, we configure the TN3270E servers within sysplex using LU name server (LUNS) and LU name requester (LUNR), as shown in Figure 2-12 on page 103. Telnet server TNLUNS31 in system SC31 that is used for centralized LU allocation is referred to as the primary LU name server. Backup LU name server TNLUNS30 is running in system SC30. Clients connect to the distributed DVIPA address of the TN3270E server TN3270A and TN3270B. Based on installation policies, the Sysplex distributor (SD) then connects directly to the best available TN3270E server. Telnet servers that accept client connections request an LU name from the LU name server instead of directly assigning an LU name.

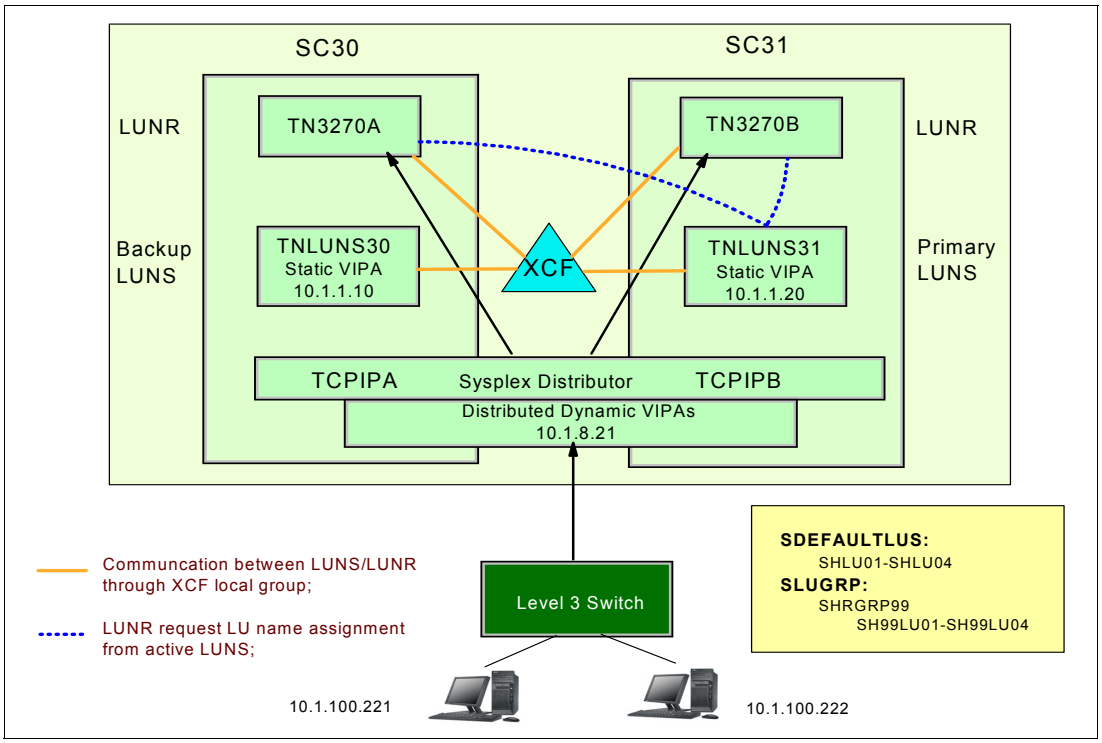


Figure 2-12 TN3270E server within the sysplex using LU name server and LU name requester

This scenario is based on the TN3270E servers within the sysplex discussed in 2.2, “TN3270E server in a single image” on page 43.

In addition to the configuration tasks for every TN3270E server instance discussed in 2.3.2, “Configuration of multiple TN3270E servers within the sysplex” on page 80, the following tasks are necessary to configure multiple TN3270E servers within sysplex using LU name server and LU name requester:

- Customize primary and backup LU name server started task procedure.
- Customize primary and backup LU name server configuration profile data set.
- Define system security for LU name server started task.
- Customize TN3270E server configuration profile for LU name requester.
- Define APPL major node for shared LUs.

Customize primary and backup LU name server started task procedure

Basically, an LU name server is a Telnet server. We can use the sample JCL in SEZAINST(EZBTNPRC).

Specify the customized profile data set name on the PROFILE DD entry and the customized tcpdata data set name on the SYSTCP DD statement in the JCL.

The JCL for the primary LU name server (LUNS) started task is shown in Example 2-73 on page 104.

Example 2-73 JCL for the LU name server primary server: TNLUNS31

```
BROWSE    SYS1.PROCLIB(TNLUNS31) - 01.02            Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
//TNLUNS31   PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=LUNS&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//TNLUNS   EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
```

The JCL for the backup LU name server started task can be modeled after the primary LU name server started task. The JCL for backup LU name server started task is shown in Example 2-74. You can also use same JCL for both started task procedures if possible, using system symbolics to provide unique names for configuration files.

Example 2-74 JCL for the LU name server backup server: TNLUNS30

```
//TNLUNS30  PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=LUNS&SYSCLONE.,TCPDATA=DATAA&SYSCLONE.
//TNLUNS   EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(&TCPDATA.)
```

Customize primary and backup LU name server configuration profile data set

The new XCFGROUP configuration statement determines the roles the TN3270E Telnet server is capable of playing and configures new parameters. This XCFGROUP statement block is on the TELNETGLOBALS statement block. The statements, their parameters, and statement syntax are discussed in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

IP address for active LU name server

In a sysplex environment, a static VIPA is the best type of IP address to use for the LU name server administrative listener socket. Dynamic VIPA can work for the LU name server administrative listener, but there are cases in which dynamic VIPA can cause confusion:

- If VIPADEFINE moves the IP address away from the LU name server, LUNRs already connected to the LU name server can continue to send requests and receive replies. However, a new LU name requester attempting to connect is directed to the new TCP/IP stack, and the LU name server are not there.
- VIPARANGE works if you can guarantee that each LU name server is supported by a different TCP/IP stack and that the LU name server is the application that creates the IP address. If not, and the creating application ends, the LU name server socket is closed. A standby LU name server on the same stack will cause a problem. The standby LU name server sets up its listener before the original LU name server finishes cleanup. When the original LU name server finishes, the new LU name server listener socket is closed.

Telnet SUBPLEX

The SUBPLEX suffix can be used to modify the name of the XCF group to join. If your environment has one or more of the following characteristics, then you need to partition the sysplex into Telnet subplexes:

- If the sysplex is partitioned into TCP/P subplexes that do not have connected routes to each other, then partition the sysplex into Telnet subplexes along the same TCP/IP boundaries.
- If your sysplex is partitioned into VTAM subplexes and the VTAMs are in different networks, then partition the sysplex into Telnet subplexes along the same VTAM network boundaries.
- If you use several Telnet ports, each with a large set of shared LU names that do not overlap and if you load balance these ports across several Telnet servers, you might want to partition the sysplex into Telnet subplexes along the Telnet port boundaries. Subplexes reduce the shared LU name management workload on the LU name server and provide operational independence of the LU name server.

LU name server configuration profile in our environment

As shown in Figure 2-12 on page 103, we define two LU name server (LUNS) configuration files, one for primary LU name server and the other for backup LU name server. If the Telnet server only acts as an LU name server, we can just use the TELNETGLOBALS statement to define it. Example 2-75 defines the primary LU name server configuration.

Example 2-75 Primary LU name server configuration profile data set: TCPIPB.TCPPARMS(LUNS31)

```
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB             1
    XCFGROUP
        JOIN XCFMONITOR 60          2
        LUNS 10.1.1.20 4444         3
        PRIMARY                     4
        RANK 101                    5
    ENDXCFGROUP
    TNSACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
ENDTELNETGLOBALS
```

In the configuration profile, we define following parameters:

1. Define the stack affinity with TCPIPB.

2. Use JOIN to control the server join an XCF group. An LU name server must join an XCF group to manage shared LU name group objects. Configure XCFMONITOR to set the frequency with which you want XCF to monitor the health of the Telnet LU name server or LU name requester.

3. Specify the LU name server on the XCFGROUP statement. Specify the IP address and port where this Telnet will listen for LU name requester connection requests when it becomes the active LU name server. Here, we use the static VIPA 10.1.1.20 and port 4444 for primary LU name server in system SC31.

4. Specify this Telnet server is a primary LU name server.

5. RANK *nnn*: Specify the RANK, 1 to 255, used at recovery time when an active LU name server fails. The standby LU name server with the highest rank will become the new LU name server.

Example 2-76 defines the backup LU name server (LUNS) configuration profile data set.

Example 2-76 Backup LU name server configuration profile data set: TCPIPA.TCPPARMS(LUNS30)

```
TELNETGLOBALS
    TCPIPJOBNAME TCPIPA            1
    XCFGROUP
        JOIN XCFMONITOR 60
        LUNS 10.1.1.10 4444        2
        BACKUP                     3
        RANK 99                    4
    ENDXCFGROUP
    TNSACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
ENDTELNETGLOBALS
```

In the configuration profile, we define following parameters:

1. Define the stack affinity with TCPIPA.
2. Specify the LU name server on the XCFGROUP statement. Specify the IP address and port where this Telnet will listen for LU name requester connection requests when it becomes the active LU name server. Here we use the static VIPA 10.1.1.10 and port 4444 for backup LU name server in system SC30.
3. Specify this Telnet server is a primary LU name server.
4. RANK *nnn*: Specify the RANK, 1 to 255, used at recovery time when an active LU name server fails. The standby LU name server with the highest rank will become the new LU name server.

Define system security for LU name server started task

Before you can start the LU name server, you need to define security for the procedure name and its associated user ID. See the TN3270 chapter in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for information about setting up security for the started task. Also, review the sample file SEZAINST(EZARACF), which contains sample security statements for this effort.

This discussion assumes RACF is the security subsystem that is used. If another security product is used, see its documentation for equivalent setup instructions.

The procedure name must be added to the RACF STARTED class and have a user ID associated with it, we use the same user ID TN3270 as other TN3270E Telnet servers for LU name server started task. The definition statement is as follows:

```
RDEFINE STARTED TNLUNS*.* STDATA(USER(TCPIP))
SETROPTS RACLIST(STARTED) REFRESH
```

Coding the started task name using the wildcard format enables you to run multiple LU name server started tasks without having to define each one separately. Their names are all spelled TNLUNS*x*, where *x* is the qualifier. They can all be assigned to the same user ID.

Customize TN3270E server configuration profile for LU name requester

A single Telnet server can support both shared and unshared LU groups. Existing unshared LU group definitions continue to be managed at the local Telnet level. In our configuration, we used only define shared LU groups.

We perform the following tasks to customize the TN3270E server configuration profile for LU name requester:

1. Specify the JOIN parameter in XCFGROUP statement block to allow Telnet server join in the Telnet XCF group.
2. Configure XCFMONITOR to set the frequency with which you want XCF to monitor the health of the Telnet LU name server or LU name requester.
3. Configure CONNECTTIMEOUT to specify the length of time that an LU name requester attempts to establish a connection with an LU name server before quiescing ports that have shared groups and before dropping connections that are waiting for an LU.
4. Configure RECOVERYTIMEOUT to specify the length of time that the LU name requester attempts to establish a connection with an LU name server in recovery, before dropping active client connections that have shared LU names assigned to them.
5. Define the share LU groups for the LU name requester.

A portion of the TN3270E configuration profile data set is shown in Example 2-77.

Example 2-77 LU name requester configuration profile data set

```
TELNETGLOBALS
    TCPIPJOBNAME TCPIPA                  1
    XCFGROUP                             2
        JOIN XCFMONITOR 60
        RECOVERYTIMEOUT 60
        CONNECTTIMEOUT 60
    ENDXCFGROUP
    TNSACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
....
ENDTELNETGLOBALS
BEGINVTAM
  PORT 23
  SDEFAULTLUS                            3
      SHLU01..SHLU04
  ENDSDEFAULTLUS
  SLUGROUP                               4
      SHRGRP99  SH99LU01..SH99LU04
  ENDSLUGROUP

  LUMAP SHRGRP99 10.1.100.221            5
....
ENDVTAM
```

In the configuration profile, we define following parameters:

1. Define the stack affinity with TCPIPA.

2. Use XCFGROUP statement block and specify JOIN to join in the XCF group.

3. Define the shared default LUs.

> **Note:** A profile can have either DEFAULT or SDEFAULT defined, but not both.

4. Define the shared group LUs.

5. Use LUMAP to define the mapping of the shared group LUs name to the workstation's IP address.

An LU name server Telnet can also be an LU name requester Telnet. You can use a portion of the configuration statement shown in Example 2-78 to define the combined LU name server (LUNS) and LU name requester (LUNR) that are running in the same Telnet server.

Example 2-78 Configuration profile for LUNS and LUNR running in same Telnet server

```
TELNETGLOBALS
    TCPIPJOBNAME TCPIPA
    XCFGROUP
       JOIN XCFMONITOR 60
       LUNS 10.1.1.10 4444            1
       BACKUP
       RANK 99
       RECOVERYTIMEOUT 60
       CONNECTTIMEOUT 60
    ENDXCFGROUP
    TNSACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
....
ENDTELNETGLOBALS
BEGINVTAM
  PORT 23
  SDEFAULTLUS
      SHLU01..SHLU04
  ENDSDEFAULTLUS
  SLUGROUP
      SHRGRP99  SH99LU01..SH99LU04
  ENDSLUGROUP

  LUMAP SHRGRP99 10.1.100.221
....
ENDVTAM
```

In this example, the number corresponds to the following information:

1. Define LU name server parameter in XCFGROUP statement to enable LU name server functions in the Telnet server.

Define APPL major node for shared LUs

The TN3270E server uses application LUs that are defined in VTAM application (APPL) major nodes to represent clients by making them look and act like VTAM terminal LUs. An APPL statement can be entered for each LU or a model APPL statement that is used. You should use a model statement to avoid all the clerical effort of maintaining a large list.

We define the APPL major node for the shared LUs as shown in Example 2-79.

Example 2-79 APPL major node for shared LUs

```
BROWSE    SYS1.VTAMLST(@TCPSLUS) - 01.02            Line 00000000 Col 001 080
 Command ===>                                          Scroll ===> CSR
******************************** Top of Data ********************************
TCPLUS&SYSCLONE. VBUILD TYPE=APPL
SH*          APPL  AUTH=NVPACE,                                          X
               EAS=1,                                                    X
               PARSESS=NO,                                               X
               SESSLIM=YES,                                              X
               MODETAB=ALLMODES
```

The actual name assigned for a connection is determined by the TN3270E server mapping rules and is assigned at the time of connection negotiation.

2.4.3 Activation and verification of LU name server and requester within sysplex

In this section, we show how to activate, verify, and manage multiple TN3270E servers within the sysplex using LU name server and LUNR. You can use a number of Telnet console commands to show the status and to control Telnet address space. Because the TN3270E Telnet server can be run only in its own address space, the keyword TELNET is no longer needed to route commands between the stack and Telnet command processors. The keyword is still supported for automation routines and operator habits. The format of the Telnet console command is as follows:

```
D TCPIP,tnproc,command,option
```

For complete details about the Telnet command, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Note: If you are running the Telnet server on multiple systems, ensure that all systems can function in a sysplex. You do not need a coupling facility, but the systems must have at least an XCF group connection. Make sure the VIPA address that is used by the LU name server is reachable and that the routing table is correct if you are using a dynamic routing method.

To verify the environment of LU name server and LU name requester within the sysplex, follow these steps:

1. Start the primary and backup LU name server
2. Start the LU name requester Telnet servers and show the LU name server and LU name requester status
3. Start Telnet sessions and verify the LU name server LU management functions
4. Control the LU name server and perform planned the LU name server takeover

Start the primary and backup LU name server

Issue the MVS START to start the primary LU name server (LUNS) TNLUNS31 on system SC31:

```
S TNLUNS31
```

Example 2-80 shows the initialization messages that display when the LU name server starts.

Example 2-80 Initialization messages when primary LU name server starts

```
 $HASP373 TNLUNS31 STARTED
 IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
 EZZ6001I TNLUNS31 SERVER STARTED
 EZZ6044I TNLUNS31 PROFILE PROCESSING BEGINNING FOR FILE 973
             TCPIPB.TCPPARMS(LUNS31)
 EZZ6045I TNLUNS31 PROFILE PROCESSING COMPLETE FOR FILE
             TCPIPB.TCPPARMS(LUNS31)
 EZZ6091I TNLUNS31 JOINED XCF GROUP EZZTLUNS                  1
*EZZ6095I TNLUNS31 LUNS CONN PENDING                          2
 EZZ6093I TNLUNS31 LUNS ACTIVE                                3
 EZZ6041I TNLUNS31 SNMP SUBAGENT INITIALIZATION COMPLETE
```

In this example, the numbers correspond to the following information:

1. The LU name server joined in the XCF group. The default group name is *EZZTLUNS*. A suffix from one to four characters can be specified in the SUBPLEX suffix operand. The suffix is right justified and overlays the end of the default name.
2. The LU name server is waiting to connect to a previously active LU name requester that is using shared LUs or that is not aware of the new LU name server. The primary LU name server becomes the active LU name server when there is no active LU name server found.
3. The primary LU name server becomes the active LU name server.

Use the display command to check the LU name server (LUNS) XCF group status, as shown in Example 2-81.

Example 2-81 Display XCF group information for primary LU name server

```
D TCPIP,TNLUNS31,XCF
EZZ6089I TNLUNS31 XCF GROUP DISPLAY 993
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60                  1
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.20..4444                                  2
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ --------------
SC31     TNLUNS31         1 P101 ACTIVE         STANDBY         3
9 OF 9 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. The XCF group name is EZZTLUNS.
2. The active LU name server listens for the LU name requester connection requests from the static VIPA 10.1.1.20 and port 4444 defined in the LU name server configuration profile data set.
3. This Telnet server acts as the ACTIVE LU name server. It is defined as primary (P) and the rank level is 101.

Now, start the backup LU name server (LUNS) TNLUNS30 on system SC30, as shown in Example 2-82.

Example 2-82 Initialization messages when backup LU name server starts

```
$HASP373 TNLUNS30 STARTED
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TNLUNS30 SERVER STARTED
EZZ6044I TNLUNS30 PROFILE PROCESSING BEGINNING FOR FILE 603
           TCPIPA.TCPPARMS(LUNS30)
EZZ6045I TNLUNS30 PROFILE PROCESSING COMPLETE FOR FILE
           TCPIPA.TCPPARMS(LUNS30)
EZZ6041I TNLUNS30 SNMP SUBAGENT INITIALIZATION COMPLETE
EZZ6091I TNLUNS30 JOINED XCF GROUP EZZTLUNS                  1
```

In this example, the number corresponds to the following information:

1. The backup LU name server joined in the XCF group.

Note: When a backup LU name server starts first, it has a JOINED state and waits for the active LU name server. After the primary LU name server starts and becomes the active LU name server, the backup LU name server can change the state from JOINED to STANDBY and act as the standby LU name server.

Use the display command to check the LU name server (LUNS) XCF group status, as shown in Example 2-83.

Example 2-83 Display XCF group information for primary and secondary LU name server

```
D TCPIP,TNLUNS30,XCF
EZZ6089I TNLUNS30 XCF GROUP DISPLAY 640
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.20..4444                                      1
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ --------------
SC30     TNLUNS30         1 B 99 STANDBY       STANDBY            2
SC31     TNLUNS31         1 P101 ACTIVE        STANDBY            3
10 OF 10 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. The active LU name server listens for the LU name requester connection requests from the static VIPA 10.1.1.20 and port 4444 defined in the LU name server configuration profile data set.
2. This Telnet server acts as the STANDBY LU name server. It is defined as backup (B) and the rank level is 99.

Start the LU name requester Telnet servers and show the LU name server and LU name requester status

Before starting LU name requester Telnet servers, you need to activate the APPL major node for the shared LU in the systems. Issue a VTAM command to activate the major node on each system where you will use shared LUs. For our environment, we used the following command:

```
V NET,ACT,ID=@TCPSLUS
```

Check the LU status to make sure that it activates successfully, as shown in Example 2-84. The status of the major node is ACTIVE, and the minor nodes are in CONCT status if no sessions have been established.

Example 2-84 APPL major node status

```
D NET,ID=@TCPSLUS,E
IST097I DISPLAY ACCEPTED
IST075I NAME = @TCPSLUS, TYPE = APPL SEGMENT 017
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
IST360I APPLICATIONS:
IST080I SH*      CONCT
IST314I END
```

Then, start the LU name requester (LUNR) Telnet servers with the MVS START command, as shown in Example 2-85.

Example 2-85 Initialization messages when LUNR starts up

```
RO SC31,S TN3270B,PROFILE=TELNB31C
 ...
 IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
 EZZ6001I TN3270B SERVER STARTED
 EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 242
             TCPIPB.TCPPARMS(TELNB31C)
 EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
             TCPIPB.TCPPARMS(TELNB31C)
 EZZ6091I TN3270B JOINED XCF GROUP EZZTLUNS                      1
 EZZ6041I TN3270B SNMP SUBAGENT INITIALIZATION COMPLETE
*EZZ6092I TN3270B LUNR PROFILE PENDING
 EZZ6003I TN3270B LISTENING ON PORT   992
 EZZ6093I TN3270B LUNR ACTIVE                                    2
 EZZ6003I TN3270B LISTENING ON PORT    23

RO SC30,S TN3270A,PROFILE=TELNA30C
 ...
 IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
 EZZ6001I TN3270A SERVER STARTED
 EZZ6044I TN3270A PROFILE PROCESSING BEGINNING FOR FILE 109
             TCPIPA.TCPPARMS(TELNA30C)
 EZZ6045I TN3270A PROFILE PROCESSING COMPLETE FOR FILE
             TCPIPA.TCPPARMS(TELNA30C)
 EZZ6041I TN3270A SNMP SUBAGENT INITIALIZATION COMPLETE
 EZZ6091I TN3270A JOINED XCF GROUP EZZTLUNS                      1
*EZZ6092I TN3270A LUNR PROFILE PENDING
 EZZ6003I TN3270A LISTENING ON PORT   992
 EZZ6093I TN3270A LUNR ACTIVE                                    2
 EZZ6003I TN3270A LISTENING ON PORT    23
```

In this example, the numbers correspond to the following information:

1. TN3270A and TN3270B joined in the XCF group EZZTLUNS.

2. The LU name requester is ACTIVE.

Use the display command to check the LU name server (LUNS) and LU name requester (LUNR) XCF group status, as shown in Example 2-86.

Example 2-86 XCF status for LU name server and LUNR

```
D TCPIP,TNLUNS31,XCF                                             1
EZZ6089I TNLUNS31 XCF GROUP DISPLAY 464
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.20..4444                                   2
                          LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ --------------
SC30     TNLUNS30         1 B 99 STANDBY        STANDBY
SC30     TN3270A          1                     ACTIVE           3
SC31     TNLUNS31         1 P101 ACTIVE         STANDBY
SC31     TN3270B          1                     ACTIVE           3
12 OF 12 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. Use the `D TCPIP,tnproc,XCF` command to display the XCF group information. Note that you can use any Telnet server proc name that is in the same XCF group to display the XCF group information.
2. The active LU name server is listening through port 4444 of SC31 static VIPA 10.1.1.20.
3. TN3270A and TN3270B act as the LU name requester and their state is ACTIVE now.

Now, use the NETSTAT command to display the connection information of the LU name server (LUNS) and LU name requester (LUNR) Telnet servers. Example 2-87 shows the connection information for both the TCPIPA and TCPIPB.

Example 2-87 Connection information of LUNS and LUNR in SC31

```
D TCPIP,TCPIPB,N,CONN
USER ID  CONN     STATE
TNLUNS31 00000653 ESTBLSH
  LOCAL SOCKET:   10.1.1.20..4444    1
  FOREIGN SOCKET: 10.1.1.20..1035
TNLUNS31 00000651 LISTEN
  LOCAL SOCKET:   10.1.1.20..4444    2
  FOREIGN SOCKET: 0.0.0.0..0
TNLUNS31 00000655 ESTBLSH
  LOCAL SOCKET:   10.1.1.20..4444    3
  FOREIGN SOCKET: 10.1.2.10..1033
TN3270B  0000061A LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
TN3270B  00000652 ESTBLSH
  LOCAL SOCKET:   10.1.1.20..1035    1
  FOREIGN SOCKET: 10.1.1.20..4444
...
D TCPIP,TCPIPA,N,CONN
USER ID  CONN     STATE
TN3270A  00000C65 ESTBLSH
  LOCAL SOCKET:   10.1.2.10..1033    3
  FOREIGN SOCKET: 10.1.1.20..4444
TN3270A  00000BFD LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
...
```

In this example, the numbers correspond to the following information:

1. The connection between TN3270B and TNLUNS31 on system SC31.
2. The active LU name server listens for LU name requester connection requests from port 4444, static VIPA 10.1.1.20.
3. The connection between TN3270B on system SC30 and TNLUNS31 on system SC31.

There is no connection between active LU name server TNLUNS31 and standby LU name server TNLUNS30. They use XCF group services to change the member information from each other. Each member of the group receives notification from XCF when another member joins or leaves the group and when any member updates its member user status field. Telnet uses the XCF member status field to communicate the roles each member is playing in the group, the state of each member's LU name requester and LU name server, the instance count of the LU name server each member currently recognizes as the controlling LU name server in the group, and various status and problem flags to which other members might need to respond.

Start Telnet sessions and verify the LU name server LU management functions

Note: In preparation for the examples in this section, we connected several clients from workstation 10.1.100.221 and 10.1.100.222 to the Dynamic VIPA 10.1.8.21. The distribution method for 10.1.8.21 port 23 is set to ROUNDROBIN for our test to show the result of LU assignment by LU name server.

Based on the LUMAP statement in Telnet server configuration profile, clients from workstation 10.1.100.221 use shared LU group SHRGRP99, and the LU name is SH99LU01 to SH99LU04. Clients from workstation 10.1.100.222 use the default shared LU definition, and the LU name is SHLU01 to SHLU04. These LU names are assigned by the active LU name server, which ensure that only one LU name requester is using the LU name.

Example 2-88 shows the connections on TN3270A; Example 2-89 shows the connections on TN3270B. Because we use ROUNDBOBIN as our distribution method, the client connects these two TN3270 Telnet servers one by one, and LU name server assigns the LU name for each client connection to ensure that it does not use duplicate LU names (1 and 2).

Example 2-88 Connections on TN3270A

```
D TCPIP,TN3270A,CONN
EZZ6064I TN3270A CONN DISPLAY 930
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000F68    ::FFFF:10.1.100.221..2576
                                   SH99LU01 SC30TS06  TAE SNX32702      1
00000F79    ::FFFF:10.1.100.222..3205
                                                      ?N?               3
00000E90    ::FFFF:10.1.100.222..3181
                                   SHLU02   SC30TS04  TAE SNX32702      2
00000E92    ::FFFF:10.1.100.222..3183
                                   SHLU04   SC30TS05  TAE SNX32702      2
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      4
------------------------------------------------------------
15 OF 15 RECORDS DISPLAYED
```

Example 2-89 Connections on TN3270B

```
D TCPIP,TN3270B,CONN
EZZ6064I TN3270B CONN DISPLAY 338
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
0000075D    ::FFFF:10.1.100.222..3180
                                   SHLU01   SC31TS04  TAE SNX32702      2
0000075F    ::FFFF:10.1.100.222..3182
                                   SHLU03   SC31TS05  TAE SNX32702      2
000007CC    ::FFFF:10.1.100.221..2577
                                   SH99LU02 SC31TS06  TAE SNX32702      1
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      3
------------------------------------------------------------
13 OF 13 RECORDS DISPLAYED
```

When all the LU names in the LU pool are used, no connection can be established. As shown 3 in Example 2-88 on page 114, the LU name server cannot assign the LU name to connection 00000F79. Figure 2-13 also shows the error message in the PCOM of the workstation.

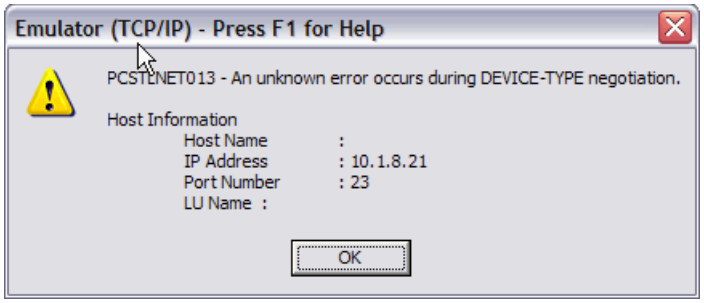


Figure 2-13 Error message in PCOM

Display commands for LU name server and LU name requester

New commands show the LU name server and LU name requester status and shared LU status. The DISPLAY OBJECT command has two Object types, SLUGRP and SPRTGRP, to support shared LU group, as shown in Example 2-90.

Example 2-90 TN3270E DISPLAY OBJECT command

```
D TCPIP,TN3270B,OBJ,TYPE=SLUGRP,ID=SHRGRP99
EZZ6083I TN3270B OBJECT DISPLAY 480
OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
----------  ------ --------- ---------------- ---------- --------
SLUGRP
 SHRGRP99        1 IPADDR    10.1.100.221                          1
                                                        --SLG---
SLUGRP: SHRGRP99
 LU STATUS                                  4 LUS TOTAL
    SH99LU01..SH99LU04..FFFFFFFN            4 LUS      1 IN USE       2
  -SH99LU02                                                           3
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      3
------------------------------------------------------------
14 OF 14 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. One connection is using shared LU group SHRGRP99, the IP address of the client is 10.1.100.221.
2. LU SH99LU01 to SH99LU04 are defined in shared LU group SHRGRP99.
3. The LU name that the active connection using is SH99LU02.

The DISPLAY LUNS OBJECT command shows summary or detail information about LU groups and where an LU name is being used. The LUNS keyword is required, and the object type can be SLUGRP, SPRTGRP, or LUS. The difference between the Classic Telnet OBJECT display and the LUNS OBJECT display is the lack of mapping information. The LU name server does not know about mapping statements and does not display Client Identifier information.

Example 2-91 on page 116 shows output for the DISPLAY LUNS OBJECT command for object type LUS.

Example 2-91 DISPLAY LUNS OBJECT command for object type LUS

```
D TCPIP,TNLUNS31,LUNS,OBJ,TYPE=LUS
EZZ6085I TNLUNS31 LUNS OBJECT 307
OBJECT      CONNS
NAME        USING   OPTIONS
----------  ------  --------
SLUGRP
 *DEFLUS*        0  --S-----
 SHRGRP99        1  --S-----                                        1
----- PORT:    23 SC30     TN3270A  PROF: 0001 CONNS:      3     2
-------------------------------------------------------------
SLUGRP
 *DEFLUS*        0  --S-----
 SHRGRP99        1  --S-----
----- PORT:    23 SC31     TN3270B  PROF: 0001 CONNS:      3
-------------------------------------------------------------
15 OF 15 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. There is one connection using LU name in shared LU group SHRGRP99 for TN3270A Telnet server.
2. There are a total of 3 sessions in TN3270A Telnet server.

Example 2-92 shows the output of the DISPLAY LUNS OBJECT command for object type SLUGRP.

Example 2-92 DISPLAY LUNS OBJECT command for object type SLUGRP

```
D TCPIP,TNLUNS31,LUNS,OBJ,TYPE=SLUGRP,ID=SHRGRP99
EZZ6085I TNLUNS31 LUNS OBJECT 311
OBJECT      CONNS
NAME        USING   OPTIONS
----------  ------  --------
SLUGRP
 SHRGRP99        1  --S-----                                          1
SLUGRP: SHRGRP99
 LU STATUS                                    4 LUS TOTAL
    SH99LU01..SH99LU04..FFFFFFFN              4 LUS      2 IN USE     2
  -SH99LU01 -SH99LU02                                                3
----- PORT:    23 SC30     TN3270A  PROF: 0001 CONNS:      3
-------------------------------------------------------------
SLUGRP
 SHRGRP99        1  --S-----
SLUGRP: SHRGRP99
 LU STATUS                                    4 LUS TOTAL
    SH99LU01..SH99LU04..FFFFFFFN              4 LUS      2 IN USE
  -SH99LU01 -SH99LU02
----- PORT:    23 SC31     TN3270B  PROF: 0001 CONNS:      3
-------------------------------------------------------------
21 OF 21 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. There is one connection using LU name in shared LU group SHRGRP99.

2. There are total 4 LUs defined in shared LU group SHRGRP99, two of them are in used.

3. The using LU name is SH99LU01 and SH99LU02.

The Where-Used option has additional information about the LUNs version. In addition to showing where an LU name is used within a profile, it also shows the status of the LU name. See Example 2-93.

Example 2-93 DISPLAY LUNS OBJECT WU example

```
D TCPIP,TNLUNS31,LUNS,OBJ,TYPE=WU,ID=SH99LU01
EZZ6085I TNLUNS31 LUNS OBJECT 364
OBJECT      CONNS
NAME        USING   OPTIONS
----------  ------  --------
SLUGRP
 SHRGRP99        1  --S-----
----- PORT:    23 SC30     TN3270A  PROF: 0001 CONNS:      3
------------------------------------------------------------
SLUGRP
 SHRGRP99        1  --S-----
----- PORT:    23 SC31     TN3270B  PROF: 0001 CONNS:      3
------------------------------------------------------------
LU: SH99LU01  STATUS: IN USE BY SC30     TN3270A                1
14 OF 14 RECORDS DISPLAYED
```

In this example, the number corresponds to the following information:

1. LU SH99LU01 is in used by TN3270A Telnet server in system SC30.

Telnet calculates statistics on both sides of each administrative connection between an LU name requester and an LU name server. Statistics are reported for the last completed statistics interval and for the weighted average of those values over the previous 10 intervals. The `DISPLAY TPCIP,tnpoc,XCF,STATS` command displays the Telnet statistics about connections to all partners, as shown in Example 2-94.

Example 2-94 D TCPIP,tnproc,XCF,STATS example

```
D TCPIP,TNLUNS31,XCF,STATS
EZZ6088I TNLUNS31 XCF STATS DISPLAY 369
    INTERVAL: 60S           PEND       RECV          SEND
 NEXT UPDATE: 43S    RTT    RCRD    TIME  RCRD    TIME  RCRD
====PARTNERS=====
SC30     TN3270A   -----   -----   ----- -----   ----- -----
            LAST:   355U      0     139U    8     408U    8
             AVG:   418U      0     136U    6     385U    6
SC31     TN3270B   -----   -----   ----- -----   ----- -----
            LAST:    85U      0      79U    8     168U    8
             AVG:    77U      0      79U    6     170U    6
11 OF 11 RECORDS DISPLAYED
```

Inactivate and activate LU names in the LU name server LU table

You can use the VARY LUNS INACT and ACT commands to inactivate and activate LU names in the LU name server LU table. The commands are similar to classic Telnet VARY INACT and ACT commands. The difference is that you must specify the LU name server instead of Telnet after the Telnet proc name to direct the command to the LU name server LU table instead of the classic or LU name requester LU table. See Table 2-2 for the RACF user profile for these VARY LUNS INACT and ACT commands.

Example 2-95 inactivates shared LU SH99LU03 in shared LU group SHRGRP99 (1). With the `DISPLAY TCPIP,tnproc,LUNS,INACTLUS` command, you can see the inactive LUs in LU name server LU table (2). The client cannot use this LU name to establish connection with TN3270 Telnet server.

Example 2-95 INACT LU in LU name server LU table

```
V TCPIP,TNLUNS31,LUNS,INACT,SH99LU03                          1
EZZ6038I TNLUNS31 COMMAND INACT SH99LU01 COMPLETE

D TCPIP,TNLUNS31,LUNS,INACTLUS
EZZ6062I TNLUNS31 LUNS INACTLUS 400
INACTIVE LUS
               SH99LU03                                       2
4 OF 4 RECORDS DISPLAYED
```

Note: You cannot use the VARY LUNS INACT command to inactivate an LU name that is in use.

Control the LU name server and perform planned the LU name server takeover

VARY commands support the LU name coordination function. The LUNS QUIESCE/RESUME pair is used to remove a standby LU name server from recovery contention or allow the LU name server definitions to be updated by an OBEYFILE. The LUNS START command causes the current standby LU name server to become ACTIVE LUNS. The VARY LUNS INACT and ACT commands inactivate and activate LU names in the LU name server LU table. See “Inactivate and activate LU names in the LU name server LU table” for details about these commands.

Authorization of these commands is controlled through the RACF profile, as shown in Table 2-2. The definition can contain a wildcard at the TELNET level (for example, MVS.VARY.TCPIP.TELNET.**).

Table 2-2 RACF profile for VARY LUNS commands

COMMANDS	RACF PROFILES
V TCPIP,tnproc,LUNS,START	MVS.VARY.TCPIP.TELNET.LUNS.START
V TCPIP,tnproc,LUNS,QUIESCE	MVS.VARY.TCPIP.TELNET.LUNS.QUIESCE
V TCPIP,tnproc,LUNS,RESUME	MVS.VARY.TCPIP.TELNET.LUNS.RESUME
V TCPIP,tnproc,LUNS,INACT,luname	MVS.VARY.TCPIP.TELNET.LUNS.INACT
V TCPIP,tnproc,LUNS,ACT,luname	MVS.VARY.TCPIP.TELNET.LUNS.ACT

In this section, we provide examples about how to use the LUNS command to control LU name server and perform planned LU name server takeover.

Use MVS START command to start TNLUNS31 again. Because TNLUNS30 is the active LU name server, TNLUNS31 acts as the standby LU name server although it has higher rank level. Example 2-96 shows the output of the `D TCPIP,tnproc,XCF` command.

Example 2-96 TNLUNS31 acts as standby LU name server

```
D TCPIP,TNLUNS31,XCF
EZZ6089I TNLUNS31 XCF GROUP DISPLAY 566
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.10..4444
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ -------------
SC30     TNLUNS30         2 B 99 ACTIVE         STANDBY
SC30     TN3270A          2                     ACTIVE     L
SC31     TNLUNS31         2 P101 STANDBY        STANDBY
SC31     TN3270B          2                     ACTIVE     L
12 OF 12 RECORDS DISPLAYED
```

With the `V TCPIP,tnproc,LUNS,START` command, we can perform the planned LU name server take over and change standby LU name server to active LU name server, as shown in Example 2-97.

Example 2-97 Planned LU name server take over through LUNS START command

```
SC31 TSU04198  V TCPIP,TNLUNS31,LUNS,START
SC31 STC04321  EZZ6038I TNLUNS31 COMMAND START  COMPLETE
SC31 STC04321 *EZZ6095I TNLUNS31 LUNS CONN PENDING
SC31 STC04321 *EZZ6094I TNLUNS31 LUNS REBUILD PENDING
SC31 STC04270 *EZZ6094I TN3270B LUNR REBUILD PENDING
SC30 STC04308 *EZZ6094I TN3270A LUNR REBUILD PENDING
SC30 STC04320  EZZ6096I TNLUNS30 LUNS STOPPED
SC31 STC04321  EZZ6093I TNLUNS31 LUNS ACTIVE
SC31 STC04270  EZZ6093I TN3270B LUNR ACTIVE
SC30 STC04308  EZZ6093I TN3270A LUNR ACTIVE
```

Example 2-98 shows the LUNS and LUNR XCF group information after the LU name server take over.

Example 2-98 XCF group information after LU name server take over

```
D TCPIP,TNLUNS31,XCF
EZZ6089I TNLUNS31 XCF GROUP DISPLAY 586
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.20..4444                                  1
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ -------------
SC30     TNLUNS30         3 B 99 STANDBY        STANDBY          2
SC30     TN3270A          3                     ACTIVE     L
SC31     TNLUNS31         3 P101 ACTIVE         STANDBY          3
SC31     TN3270B          3                     ACTIVE     L
12 OF 12 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. The active LU name server listens for the LU name requester connection request from port 4444, static VIPA 10.1.1.20 on system SC31.

2. TNLUNS30 changes from ACTIVE mode to STANDBY mode.

3. TNLUNS31 changes from STANDBY mode to ACTIVE mode.

You can use the `V TCPIP,tnproc,LUNS,QUIESCE` command to change the LU name server from the STANDBY and JOIN states to the QUIESCE state. An LU name server in QUIESCE state is ineligible to participate in recovery scenarios or to start by VARY START command. Also, an LU name server must be in QUIESCE state to make a configuration change using the `VARY TCPIP,tnproc,OBEYFILE,datasetname` command. See Example 2-99.

Example 2-99 LUNS QUIESCE command scenario

```
V TCPIP,TNLUNS30,LUNS,QUIESCE                                          1
EZZ6038I TNLUNS30 COMMAND LUNS QUIESCE  COMPLETE

D TCPIP,TNLUNS30,XCF
EZZ6089I TNLUNS30 XCF GROUP DISPLAY 471
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.20..4444
                            LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------- --------------
SC30     TNLUNS30         3 B 99 QUIESCE        STANDBY              2
SC30     TN3270A          3                     ACTIVE     L
SC31     TNLUNS31         3 P101 ACTIVE         STANDBY
SC31     TN3270B          3                     ACTIVE     L
12 OF 12 RECORDS DISPLAYED

V TCPIP,TNLUNS30,O,TCPIPA.TCPPARMS(LUNS30)                             3
EZZ6044I TNLUNS30 PROFILE PROCESSING BEGINNING FOR FILE 489
            TCPIPA.TCPPARMS(LUNS30)
EZZ6045I TNLUNS30 PROFILE PROCESSING COMPLETE FOR FILE
            TCPIPA.TCPPARMS(LUNS30)
EZZ6038I TNLUNS30 COMMAND OBEYFILE  COMPLETE
```

In this example, the numbers correspond to the following information:

1. Use LUNS QUIESCE command to change the standby LU name server to quiesce state.

2. The XCF group display command output shows that the LU name server is in QUIESCE state now.

3. Now, you can use the OBEYFILE command to updated LU name server definitions.

To change the LU name server from the QUIESCE state to the STANDBY state, use the LUNS RESUME command, as shown in Example 2-100 on page 121.

Example 2-100 LUNS RESUME command scenario

```
V TCPIP,TNLUNS30,LUNS,RESUME
EZZ6038I TNLUNS30 COMMAND LUNS RESUME  COMPLETE

D TCPIP,TNLUNS30,XCF
EZZ6089I TNLUNS30 XCF GROUP DISPLAY 506
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.20..4444
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ -------------
SC30     TNLUNS30         3 B 99 STANDBY       STANDBY
SC30     TN3270A          3                    ACTIVE     L
SC31     TNLUNS31         3 P101 ACTIVE        STANDBY
SC31     TN3270B          3                    ACTIVE     L
12 OF 12 RECORDS DISPLAYED
```

Now, stop all LU name server Telnet servers with the MVS STOP command. When the last LU name server is stopped, LU name requester Telnet servers have lost contact with LU name server and are in rebuild pending status. The EZZ6094I message is non-scrollable until the LU name requester completes recovery with a new LU name server, as shown by 1 in Example 2-101.

Example 2-101 Shutdown message of the last LU name server

```
 P TNLUNS31
 EZZ6008I TNLUNS31 STOPPING
 EZZ6096I TNLUNS31 LUNS STOPPED
 EZZ6011I TN3270A BPX1CON FAILED, RC = 00000461 RSN = 74520442
*EZZ6094I TN3270A LUNR REBUILD PENDING                      1
 EZZ6035I TN3270B DEBUG TASK   EXCEPTION 805
   TASK: LUNR                   MOD: EZBTXLUR
   RCODE: 102B-00  Socket initialization failed.  No retry.
   PARM1: 0000008C PARM2: 76697242 PARM3: BPX1SND
*EZZ6094I TN3270B LUNR REBUILD PENDING                      1
 EZZ6009I TNLUNS31 SERVER STOPPED
 IEF352I ADDRESS SPACE UNAVAILABLE
 $HASP395 TNLUNS31 ENDED
```

The XCF group information is shown in Example 2-102.

Example 2-102 XCF group information for Telnet servers without LU name server

```
D TCPIP,TN3270A,XCF
EZZ6089I TN3270A XCF GROUP DISPLAY 597
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: **N/A**                                      1
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ -------------
SC30     TN3270A          4                    RECOVER C RL       2
SC31     TN3270B          4                    RECOVER C RL       2
10 OF 10 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. No LU name server listens for the LU name requester connection request.
2. The LU name requesters are in RECOVER STATE:
   - C means the LU name requester is trying to connect to the LU name server but has not succeeded yet.
   - R means the LU name requester has not completed the rebuild process during LU name server recovery.
   - L means the LU name requester has assigned LU names, allocated from the LU name server, to client connections. The LU name requester is using shared LU names. It is considered as a critical LU name requester.

The existing client connections are not impacted by the LU name server failure, but the new client connection cannot be established successfully and the request is in pending status. The client connect requests is accepted by the LU name requester, but when the LU name requester tries to request the LU name from the LU name server, this request is pending and waits until the LU name server starts. The `D TCPIP,tnproc,CONN` command shows the connection status when we tried to start a new session from the workstation, as shown in Example 2-103.

Example 2-103 Display connection status with D TCPIP,TN3270A,CONN command

```
D TCPIP,TN3270A,CONN
EZZ6064I TN3270A CONN DISPLAY 275
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00001932    ::FFFF:10.1.100.221..4948
                                   SH99LU02 SC30TS02  TAE SNX32702     1
00001974    ::FFFF:10.1.100.222..1440
                                   SHLU02   SC30TS04  TAE SNX32702     1
00001984    ::FFFF:10.1.100.221..4950
                                   *LUNSREQ           ?N?              2
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      3
------------------------------------------------------------
13 OF 13 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. Existing client connections are not impacted by the LU name server failure.
2. LUNAME *LUNSREQ means that the connection at an LU name requester is waiting for a reply from the LU name server.

Use the connection ID to display the detail information for the new connection, as shown in Example 2-104.

Example 2-104 Display connection status with D TCPIP,TN3270A,CONN,CONN= command

```
D TCPIP,TN3270A,CONN,CONN=1984,DET
EZZ6065I TN3270A CONN DISPLAY 296
  CONNECTED: 18:10:43  09/29/2010  STATUS: NEGOTIATE IN PROGRESS    1
  CLIENT IDENTIFIER FOR CONN: 00001984   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.221..4950
    DESTIP..PORT: ::FFFF:10.1.8.21..23
    LINKNAME: VIPLOA010815
```

```
  PORT:    23 QUAL: NONE
    AFFINITY: TCPIPA
    STATUS: ACTIVE   BASIC
    ACCESS: NON-SECURE
  PROTOCOL: NEGOTIAT          DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
    OPTIONS: E-------   3270E FUNCTIONS: *N/A*
                       NEWENV FUNCTIONS: --
  LUNAME: *LUNSREQ                                                    2
  APPL: **N/A**
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED:           APPL SPECIFIED:
  MAPPING TYPE:  CONN IDENTIFIER
                       OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  IP ::FFFF:10.1.100.221
                       SHRGRP99                   --S-G---
  ...
```

In this example, the numbers correspond to the following information:

1. The STATUS field shows the connection is pending for negotiation.
2. LUNAME *LUNSREQ means that the connection at an LU name requester is waiting for a reply from the LU name server.

Use the MVS START command to start the LU name server again. When the LU name server starts and joins the XCF group, it receives updated information from all LUNRs, rebuilds the shared LU table, and becomes the active LU name server.

When the LU name requester cannot communicate with the LU name server, client connections are stuck in negotiation. If the LU name requester can re-establish its connection to the LU name server within the CONNECTTIMEOUT period, these client connections can be established, as shown in Example 2-105.

Example 2-105 Connection established for pending request

```
D TCPIP,TN3270A,CONN
EZZ6064I TN3270A CONN DISPLAY 313
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00001932    ::FFFF:10.1.100.221..4948
                                   SH99LU02 SC30TS02  TAE SNX32702
00001974    ::FFFF:10.1.100.222..1440
                                   SHLU02   SC30TS04  TAE SNX32702
00001984    ::FFFF:10.1.100.221..4950
                                   SH99LU01 SC30TS01  TAE SNX32702      1
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      3
------------------------------------------------------------
13 OF 13 RECORDS DISPLAYED
```

In this example, the number corresponds to the following information:

1. The pending connection 00001984 established successfully.

2.4.4 Scenario: LU name server automated takeover when active name server fails

In this section, we stop the active LU name server (LUNS) to see how the standby LU name server takes over the active LU name server role. When an active LU name server fails, the following sequence of events automatically occurs:

1. All the LU name servers that are in standby mode examine the list of LU name servers that are in standby mode.
2. The standby LU name server that has the highest configured rank, regardless of whether that LU name server was started as a primary or backup LU name server, becomes the active LU name server automatically.

   If several standby LU name servers have the same rank, then all those servers attempt to take over. The race winner becomes the new active LU name server and the others return to standby.

Before we stop the active LU name server, we display the current LU name server and LU name requester (LUNR) status and shared LU status, as shown in Example 2-106.

Example 2-106 Command output for LUNS and LUNR status and shared LU status

```
D TCPIP,TNLUNS31,XCF
EZZ6089I TNLUNS31 XCF GROUP DISPLAY 157
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.20..4444                                          1
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------- --------------
SC30     TNLUNS30        11 B 99 STANDBY        STANDBY                 2
SC30     TN3270A         11                     ACTIVE     L            3
SC31     TNLUNS31        11 P101 ACTIVE         STANDBY                 4
SC31     TN3270B         11                     ACTIVE     L            3
12 OF 12 RECORDS DISPLAYED

D TCPIP,TNLUNS31,LUNS,OBJ,TYPE=LUS
EZZ6085I TNLUNS31 LUNS OBJECT 186
OBJECT      CONNS
NAME        USING   OPTIONS
----------  ------  --------
SLUGRP
 *DEFLUS*        1  --S-----
 SHRGRP99        2  --S-----
----- PORT:    23 SC30     TN3270A  PROF: 0001 CONNS:      3           5
------------------------------------------------------------
SLUGRP
 *DEFLUS*        2  --S-----
 SHRGRP99        1  --S-----
----- PORT:    23 SC31     TN3270B  PROF: 0001 CONNS:      3           5
------------------------------------------------------------
15 OF 15 RECORDS DISPLAYED

D TCPIP,TN3270B,CONN                                                    6
EZZ6064I TN3270B CONN DISPLAY 192
         EN                                          TSP
```

```
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000055    ::FFFF:10.1.100.221..1312
                                   SH99LU04 SC31TS03  TAE SNX32702
00000057    ::FFFF:10.1.100.222..1583
                                   SHLU01   SC31TS04  TAE SNX32702
0000005E    ::FFFF:10.1.100.222..1585
                                   SHLU03   SC31TS05  TAE SNX32702
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      3
------------------------------------------------------------
13 OF 13 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. The active LU name server is listening for an LU name requester connection request from port 4444, static VIPA 10.1.1.20 in system SC31.

2. TNLUNS30 acts as the standby LU name server with rank 99.

3. LUNR TN3270A and TN3270B have a shared LU allocated, which is indicated by the `L` in the STATUS column.

4. TNLUNS31 acts as the active LU name server with a rank of 101.

5. There are 3 active connections for TN3270A and TN3270B.

6. Detail information for TN3270B connections.

Now, we issue the MVS STOP command to stop the active LU name server. Example 2-107 shows the messages.

Example 2-107 Stop active LU name server messages

```
SC31 TSU04974  P TNLUNS31
SC31 STC04432  EZZ6008I TNLUNS31 STOPPING
SC31 STC04432  EZZ6096I TNLUNS31 LUNS STOPPED
SC30 STC04308  EZZ6011I TN3270A BPX1CON FAILED, RC = 00000468 RSN = 76630291 1
SC31 STC04270  EZZ6011I TN3270B BPX1CON FAILED, RC = 00000461 RSN = 74520442 1
SC31 STC04270 *EZZ6094I TN3270B LUNR REBUILD PENDING                         2
SC30 STC04308 *EZZ6094I TN3270A LUNR REBUILD PENDING                         2
SC30 STC04431 *EZZ6095I TNLUNS30 LUNS CONN PENDING                           3
SC30 STC04431 *EZZ6094I TNLUNS30 LUNS REBUILD PENDING                        2
SC30 STC04431  EZZ6093I TNLUNS30 LUNS ACTIVE                                 4
SC30 STC04308  EZZ6093I TN3270A LUNR ACTIVE                                  4
SC31 STC04270  EZZ6093I TN3270B LUNR ACTIVE                                  4
SC31 STC04432  EZZ6009I TNLUNS31 SERVER STOPPED
SC31 STC04432  IEF352I ADDRESS SPACE UNAVAILABLE
SC31 STC04432  $HASP395 TNLUNS31 ENDED
```

In this example, the numbers correspond to the following information:

1. TN3270A and TN3270B lost contact with the active LU name server.

2. TN3270A and TN3270B are in rebuild status until recovery completes with a new LU name server. TNLUNS30 is in rebuild status until all critical LUNRs complete recovery.

3. TNLUNS30 takes over and is waiting for critical LUNRs to complete the rebuild.

4. TNLUNS30 becomes the active LU name server. TN3270A and TN3270B become the active LU name requester.

The LU name server (LUNS) and LU name requester (LUNR) XCF group information during the automatic take over is shown in Example 2-108.

Example 2-108 XCF group information during LUNS takeover

```
D TCPIP,TNLUNS30,XCF
EZZ6089I TNLUNS30 XCF GROUP DISPLAY 316
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.10..4444
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ --------------
SC30     TNLUNS30        12 B 99 RECOVER CR     STANDBY             1
SC30     TN3270A         12                     RECOVER    L
SC31     TN3270B         12                     RECOVER C RL        2
11 OF 11 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. LUNS TNLUNS30 is in the RECOVER state:

- `C` status means that the LU name server is waiting, during recovery, for a connection from a critical LU name requester that is using allocated shared LU names or that an LU name requester is at the wrong LU name server count.
- `R` status means that the LU name server is waiting for rebuild information from one or more LUNRs in recovery.

2. LUNR TN3270A and TN3270B are in RECOVER state:

- `R` status means that the LU name requester has not completed the rebuild process during LU name server recovery.
- `L` status means that the LU name requester has assigned LU names, allocated from the LU name server, to client connections. The LU name requester is using shared LU names. It is considered as a critical LU name requester.

After the recovery and the LU name server (LUNS) and the LU name requester (LUNR) are active, XCF group information is shown in Example 2-109.

Example 2-109 XCF group information after LU name server takeover

```
D TCPIP,TNLUNS30,XCF
EZZ6089I TNLUNS30 XCF GROUP DISPLAY 320
GROUP NAME: EZZTLUNS CONNECTTIMEOUT:        60
XCFMONITOR:       60 RECOVERYTIMEOUT:       60
LUNS LISTENER: 10.1.1.10..4444                                      1
                           LUNS--------------- LUNR----------
MVSNAME  TNNAME   PDMON CTR RANK STATE   STATUS STATE   STATUS
-------- -------- ----- --- ------------------ --------------
SC30     TNLUNS30        12 B 99 ACTIVE         STANDBY             2
SC30     TN3270A         12                     ACTIVE     L        3
SC31     TN3270B         12                     ACTIVE     L
11 OF 11 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

1. The active LU name server is listening for an LU name requester connection request from port 4444, static VIPA 10.1.1.10 of system SC30, which we defined in the configuration profile data set.
2. TNLUNS30 acts as ACTIVE LUNS.
3. LUNR TN3270A and TN3270B are ACTIVE with shared LUs allocated.

The display command output of the LUNS object and TN3270 Telnet server connections are the same as before. The takeover procedure is transparent to existing client connections. See Example 2-110.

Example 2-110 Command output for share LUNS and Telnet server connection after takeover

```
D TCPIP,TNLUNS30,LUNS,OBJ,TYPE=LUS
EZZ6085I TNLUNS30 LUNS OBJECT 414
OBJECT      CONNS
NAME        USING   OPTIONS
----------  ------  --------
SLUGRP
 *DEFLUS*        1  --S-----
 SHRGRP99        2  --S-----
----- PORT:    23 SC30     TN3270A  PROF: 0001 CONNS:      3
------------------------------------------------------------
SLUGRP
 *DEFLUS*        2  --S-----
 SHRGRP99        1  --S-----
----- PORT:    23 SC31     TN3270B  PROF: 0001 CONNS:      3
------------------------------------------------------------
15 OF 15 RECORDS DISPLAYED

D TCPIP,TN3270B,CONN
EZZ6064I TN3270B CONN DISPLAY 350
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000055    ::FFFF:10.1.100.221..1312
                                    SH99LU04 SC31TS03  TAE SNX32702
00000057    ::FFFF:10.1.100.222..1583
                                    SHLU01   SC31TS04  TAE SNX32702
0000005E    ::FFFF:10.1.100.222..1585
                                    SHLU03   SC31TS05  TAE SNX32702
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      3
------------------------------------------------------------
13 OF 13 RECORDS DISPLAYED
```

2.5 TN3270E server in a single image using SHAREACB

This section describes the steps to modify our single image scenario created in 2.2, “TN3270E server in a single image” on page 43 to implement SHAREACB statement to reduce the amount of CSA storage allocated by TN3270e Server.

To implement this scenario, we follow these steps:

1. Overview of SHAREACB utilization
2. Configuration of the TN3270E server with SHAREACB option
3. Activation of the TN3270E server
4. Verification of the TN3270E server with SHAREACB defined

2.5.1 Overview of SHAREACB utilization

Telnet server configurations that support a large number of connections can place a high demand on ECSA storage. To reduce this demand, configure the Telnet server to enable multiple Telnet LUs to share an ACB.

To configure the Telnet server to enable LUs to share an ACB, specify SHAREACB in the TELNETGLOBALS statement block. Replace any predefined (static) VTAM APPL definitions that are being used to represent Telnet LUs with corresponding model application program definitions; the latter are required when using the SHAREACB function.

In this scenario, we defined the SHAREACB statement within the TELNETGLOBALS parameter. The APPL major node to represent the Telnet connections must be defined as a model APPL, The relationship between the TN3270E server, the TN3270 client, and the TCP/IP stack is illustrated in Figure 2-14.

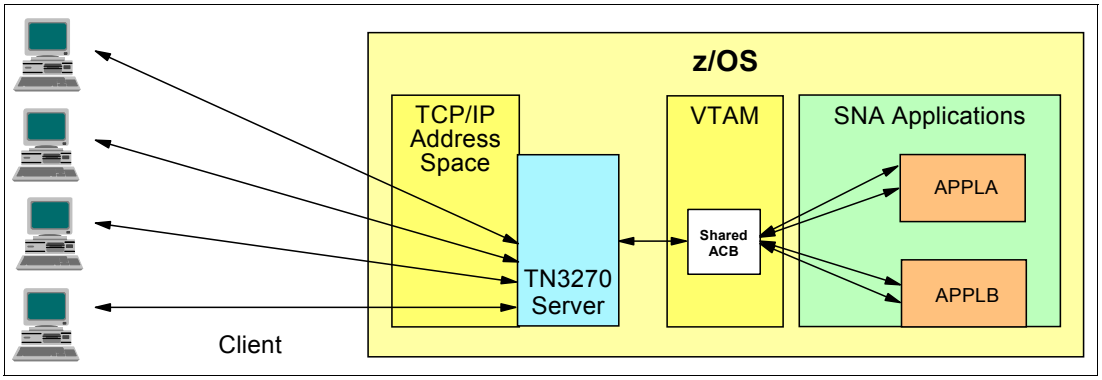


Figure 2-14 TN3270E Telnet server using SHAREACB option

As shown in Figure 2-14, with SHAREACB option defined, all Telnet connections will establish their sessions with SNA applications through a single shared ACB.

In our scenario we used the TCP/IP stack *TCPIPB* in LPAR SC31. The TN3270E started task name is *TN3270B*.

2.5.2 Configuration of the TN3270E server with SHAREACB option

Perform the following tasks to change the TN3270E server to use a shared ACB:

1. Customize a VTAM APPL model major node for the TCP/IP LU names.
2. Customize the TN3270 PROFILE to include the SHAREACB option.
3. Change the procedure TN3270B to use correct profile.

Customize a VTAM APPL model major node for the TCP/IP LU names

You can find a sample VTAM APPL major node in SEZAINST(IVPLU). See *z/OS Communications Server: IP Configuration Guide*, SC31-8775 for an in-depth discussion about how to prepare the VTAM LU definitions for the TN3270E server.

To use a VTAM shared ACB control block, we must create a model APPL statement. for further considerations about how to configure a model APPL major node, see *z/OS Communications Server: SNA Resource Definition*, SC31-8778.

We created a new member in the *SYS1.VTAMLST*, *TN3270B.* Our model statement uses an asterisk (*) in the name as a wild card. as shown in Example 2-111.

Example 2-111 Member TN3270B of SYS1.VTAMLST

```
EDIT       SYS1.VTAMLST(TN3270B) - 01.05                        Columns 00001
*
*  APPL MODEL MAJOR NODE FOR SHAREACB OPTION IN TN3270E SERVER
*
TN3270B  VBUILD TYPE=APPL
SC&SYSCLONE.B* APPL AUTH=NVPACE,EAS=1,PARSESS=NO,SESSLIM=YES,          *
MODETAB=ALLMODES
```

The &SYSCLONE value (30, 31, and 32 in our test environment) is used to generate the label on the VBUILD statement, and the &SYSNAME value (SC30, SC31, SC32 in our test environment) is used to generate the actual APPL model name.

To activate the application definition major node automatically, include it in ATCCONxx.

Customize the TN3270 PROFILE to include the SHAREACB option

To include the SHAREACB parameter, we used the TN3270 profile located in*TCPIPB.TCPPARMS(TN3270B).*

A portion of the TN3270 configuration profile data set is shown in Example 2-112.

Example 2-112 TN3270B configuration profile (TN3270B) for Shared ACB utilization

```
===>                                                        Scroll ===>
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
    SHAREACB                                  1
    TNSACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
...
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
;
ENDTELNETGLOBALS
...
BEGINVTAM
  PORT 23
```

```
  DEFAULTLUS
      SC&SYSCLONE.B01..SC&SYSCLONE.B99       2
  ENDDEFAULTLUS
...
ENDVTAM
 ...
```

We changed these statements in the config file shown in Example 2-112 on page 129:

- The SHAREACB parameter is part of the TELNETGLOBALS/ENDTELNETGLOBALS statement block as seen in **1**.
- We also change the DEFAULTLUS statement block to define the range of LUs we created in the Model APPL major node in VTAM as seen in **2**.

These are the only changes we had to do on the TN3270E server to use the shared ACB on VTAM to reduce the common and private storage utilization.

For complete detailed listings of the started task procedures and profiles that we used for this scenario, see Appendix C, “Configuration files: TN3270E stand-alone scenario” on page 415.

Change the procedure TN3270B to use correct profile

Next, we changed the procedure *TN3270B* located in *SYS1.PROCLIB(TN3270B)* to use our new profile, *TCPIPB.TCPPARMS(TN3270B)* as shown in Example 2-113.

Example 2-113 JCL for the TN3270E server: TN3270

```
BROWSE    SYS1.PROCLIB(TN3270B) - 01.00             Line 00000000 Col 001 080
//TN3270B   PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=TN3270B,TCPDATA=DATAB&SYSCLONE.
//TN3270B  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
```

2.5.3 Activation of the TN3270E server

To start the TN3270E started task, issue the MVS **START** command or automate it with an automation package:

```
S TN3270B
```

The initialization messages are shown in Example 2-114.

Example 2-114 Telnet server initialization message

```
S TN3270B
 $HASP100 TN3270B  ON STCINRDR
 IEF695I START TN3270B  WITH JOBNAME TN3270B  IS ASSIGNED TO USER
 TCPIP   , GROUP TCPGRP
 $HASP373 TN3270B  STARTED
 IEF196I IEF237I DC63 ALLOCATED TO SYS00109
 IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
 EZZ6001I TN3270B SERVER STARTED
```

```
 EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 387
            TCPIPB.TCPPARMS(TN3270B)
 EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
            TCPIPB.TCPPARMS(TN3270B)
 IEF196I IEF285I   TCPIP.SEZALOAD                                KEPT
 IEF196I IEF285I   VOL SER NOS= Z1CRB1.
 EZZ6041I TN3270B SNMP SUBAGENT INITIALIZATION COMPLETE
EZZ6003I TN3270B LISTENING ON PORT    23
```

2.5.4 Verification of the TN3270E server with SHAREACB defined

To verify that the parameter is active in our TN3270 server, we used an emulation window (using IBM PCOMM) to establish a session with our TSO, checked WHICH luname we used, and executed the command **D NET,ID=luname,scope=all** with the results shown in Example 2-115.

Example 2-115 PartialD NET,ID=luname,scope=all results

```
D NET,ID=SC31B02,E
IST097I DISPLAY ACCEPTED
IST075I NAME = USIBMSC.SC31B02, TYPE = DYNAMIC APPL 508
IST486I STATUS= ACT/S, DESIRED STATE= ACTIV
IST1447I REGISTRATION TYPE = CDSERVR
IST1629I MODSRCH = NEVER
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=ALLMODES USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=***NA*** USS LANGTAB=***NA***
IST1632I VPACING =  7
IST1938I APPC = NO
IST597I CAPABILITY-PLU ENABLED  ,SLU ENABLED  ,SESSION LIMIT 00000001
IST231I APPL MAJOR NODE = TN3270B
IST1425I DEFINED USING MODEL SC31B*
IST2348I ACTIVATED BY SHARED ACB K0017-04            1
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = TN3270B, STEPNAME = TN3270B, DSPNAME = IST28070
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = 0 MAXIMUM = 0
IST1669I IPADDR..PORT 10.1.100.221..4572
IST171I ACTIVE SESSIONS = 0000000001, SESSION REQUESTS = 0000000000
```

Message IST2348I shows that the LU created from our model APPL is using Share ACB 1

2.6 TN3270 support of TSO logon reconnect

TN3270 support of TSO logon reconnect, in conjunction with TSO/E, allows you to use logon reconnect when the LOGONHERE parameter is correctly defined in SYS1.PARMLIB member IKJTSOxx. See *z/OS MVS Initialization and Tuning Reference*, SA22-7592, and *z/OS TSO/E Customization*, SA22-7783, for coding details.

With TSO logon reconnect, in the case where a TN3270 client can lose a TCP connection with the TN3270 server but the SNA session remains active, you can recover from a lost TN3270 connection and reconnect from another TN3270 connection. You can also use logon reconnect to take over a perfectly healthy TSO session from another TN3270 connection. If you share TSO user IDs with others, others can take over your TSO session while you are working.

When you reconnect from another session, you must select the Reconnect option on the bottom of the TSO logon panel, as shown in Figure 2-15.

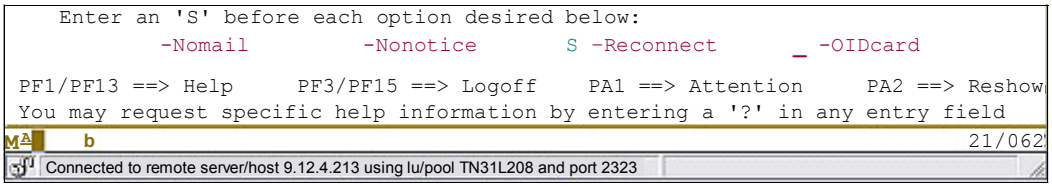


Figure 2-15 Selecting the Reconnect option

When you reconnect, the IKT00300I message displays. Then, the screen of the previous session displays. The previous session disconnects. The reconnected TN3270 connection then continues from where the previous connection was, even in the middle of an ISPF edit.

Note: The default value for LOGONHERE is set to `ON` in SYS1.PARMLIB member IKJTSOxx. Therefore, there are no tasks to enable this function. It is enabled automatically. If you want to disable it, set LOGONHERE to `OFF`.

2.7 Problem determination for the TN3270E servers

This section presents methods that you can use when performing problem determination for the TN3270 server:

- Review the definition statements within the profile
- Use TCP/IP and Telnet commands
- Use the MSG07 statement in the TN3270 profile
- Use SMF records to capture TN3270 connection activity
- Use trace data
- Tips for multiple TN3270E servers in a Parallel Sysplex environment
- Tips for LU name server and LU name requester diagnosis

2.7.1 Review the definition statements within the profile

If multiple TN3270E servers are used (for example, multiple TCP/IP stacks in a sysplex environment) and if you do not use the LU name server to coordinate the LU name in a sysplex, ensure that each server uses unique LU names. Otherwise, the second server that

uses the same LU name cannot establish a session. Either the OPEN ACB request will fail, or the cross-domain session request will fail.

Note: Use the LU name server to centralize LU name allocation and to ensure that there are no duplicate LU name assignments among the group of Telnet servers, LU name requester. See 2.4, “Multiple TN3270E servers using LU name server and LU name requester” on page 95 for detailed information about LU name server and LU name requester.

The default PPT entry sets the TN3270E server to *non-cancelable*. As a non-cancelable application, the server should not be started automatically by a TCP/IP stack using the AUTOLOG function. If the TCP/IP stack is recycled, the following messages are issued repeatedly:

```
EZZ0621I AUTOLOG FORCING TN3270B, REASON: TCP/IP HAS BEEN RESTARTED
CANCEL TN3270B,A=0075
IEE838I TN3270B           NON-CANCELABLE - ISSUE FORCE ARM
```

To prevent this, specify NOAUTOLOG on the PORT reservation statement as follows:

```
PORT 23 TCP TN3270B NOAUTOLOG
```

2.7.2 Use TCP/IP and Telnet commands

A number of TCP/IP and Telnet commands can be used to assist with TN3270 server problem determination. We discuss the more popular methods in this section.

Use OBEYFILE to turn selected debug options on and off

You can turn on or off TN3270-specific debug messages to diagnose TN3270 problems. Several types of debug messages are available.

- Summary messages indicate important status changes.
- Detail messages indicate a problem was detected.
- General messages indicate an important event.
- Trace messages show data to and from the client, and to and from VTAM.
- Flow messages show entry into and exit from modules.

Message generation is controlled with the DEBUG statement. The statement can be specified in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. A connection must have the DEBUG statement mapped to it for messages to be generated. Whenever a DEBUG message is generated, a CTRACE entry is also generated.

The format of the DEBUG statement is shown in Example 2-116. The parameters can involve subparameters, which we do not show in our examples. For details of the complete syntax, see the DEBUG statement description in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For comprehensive suggestions on the use of the DEBUG statement, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Example 2-116 DEBUG statement format

```
DEBUG parameter1  parameter2
```

As shown in Table 2-3, the DEBUG statement has two parameters: the level of tracing and the destination for trace messages. The two parameters can be specified in any combination. Note that the table does *not* imply a one-to-one relationship.

Table 2-3 DEBUG statement parameters

Trace level	Trace destination
OFF	CONSOLE (default for exception)
EXCEPTION	JOBLOG (default for summary, detail, trace)
SUMMARY	CTRACE (default for flow, profile)
DETAIL	
TRACE	
FLOW	
PROFILE	

Note: You can specify the PROFILE parameter only in TELNETGLOBALS. It has no effect on the other DEBUG statements.

If DEBUG messages are being used primarily for problem diagnosis, the VARY TCPIP,,OBEYFILE command can be used to keep the number of messages low. Bring up TN3270 initially without DEBUG coded. When a problem appears, issue a VARY TCPIP,,OBEYFILE command for a TN3270 profile that includes the DEBUG statement. Only new connections to the new profile will produce messages. After data is obtained, issue another VARY TCPIP,,OBEYFILE command for a TN3270 profile that omits the DEBUG statement or specifies DEBUG OFF.

If the client identifier of the client having the problem is known (perhaps the client's current IP address), include DEBUG in a PARMSGROUP statement. Then, using PARMSMAP, map that group to the client. Debug messages for only that client will be issued. After data is obtained, issue another VARY TCPIP,,OBEYFILE command for a TN3270 profile that omits the DEBUG statement or specifies DEBUG OFF.

An example of the profile statements and obeyfile command for activating the trace just suggested is shown in Example 2-117. Assume the profile member name for turning the trace on is TELNDBON. Note that, to avoid changing anything else in the current profile (TELNB31A), TELNDBON is a copy of TELNB31A, with these three statements added.

Example 2-117 Preparing a DEBUG TRACE statement for OBEYFILE command

```
profile statements in member TELNDBON
...
PARMSGROUP DEBUGIT  DEBUG TRACE JOBLOG ENDPARMSGROUP
IPGROUP DEBUGIPGRP 10.1.100.221
                   10.1.100.222
                   10.1.100.223
                   10.1.100.224
ENDIPGROUP

PARMSMAP DEBUGIT  IPGRP,DEBUGIPGRP
...
```

```
V TCPIP,TN3270B,O,TCPIPB.TCPPARMS(TELNDBON)
EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 114
           TCPIPB.TCPPARMS(TELNDBON)
EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
           TCPIPB.TCPPARMS(TELNDBON)
EZZ6018I TN3270B PROFILE UPDATE COMPLETE FOR PORT    23
EZZ6038I TN3270B COMMAND OBEYFILE  COMPLETE
```

When we display the CLIDs and OBJects for port 23, we see the debug options that were just set, as shown in Example 2-118.

Example 2-118 Display profile showing debug options ON

```
D TCPIP,TN3270B,CLID,PORT=23,PROFILE=ALL,DET,MAX=*
EZZ6081I TN3270B CLIENTID DISPLAY 119
CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
------------------ ------ --------- -------- ---------- --------
IPGRP
  DEBUGIPGRP
                         0 PARMSGRP  DEBUGIT             --------
NULL
  NULL
                         0 DEFAPPL   SC31TS              --------
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      0
------------------------------------------------------------
13 OF 13 RECORDS DISPLAYED

D TCPIP,TN3270B,OBJ,PORT=23,PROFILE=ALL,DET,MAX=*
EZZ6083I TN3270B OBJECT DISPLAY 122
OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
----------  ------ --------- ---------------- ---------- --------
ARAPPL
 SC*             0
                                                         -A------
 NVAS*           0
                                                         -A------
                                                       5 ----Q---
 TSO*            0
                                                         -A-D----
 *               0
                                                         -A------
DEFAPPL          0
                                                         -I------
DEFAPPL
 SC31TS          0 NULL      NULL
                                                         --------
LUGRP
 *DEFLUS*        0
                                                         --------
PARMSGRP
 DEBUGIT         0 IPGRP     DEBUGIPGRP
                                                         --------
 *DEFAULT          -------NO MAPPING---------
                                                         --------
```

```
 *TGLOBAL          -------NO MAPPING---------
                                                        --------
 *TPARMS           -------NO MAPPING---------
                                                        --------
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      0
------------------------------------------------------------
34 OF 34 RECORDS DISPLAYED
```

The profile statements and obeyfile command for deactivating the trace are shown in Example 2-119. Assume the profile member name for turning the trace off is TELNDBOF.

Example 2-119 Preparing a DEBUG EXCEPTION statement for OBEYFILE command

```
profile statements in member TELNDBOF
...
PARMSGROUP DEBUGIT  DEBUG EXCEPTION ENDPARMSGROUP
IPGROUP DEBUGIPGRP 10.1.100.221
                   10.1.100.222
                   10.1.100.223
                   10.1.100.224
ENDIPGROUP

PARMSMAP DEBUGIT  IPGRP,DEBUGIPGRP
...

V TCPIP,TN3270B,O,TCPIPB.TCPPARMS(TELNDBOF)
EZZ6044I TN3270B PROFILE PROCESSING BEGINNING FOR FILE 125
            TCPIPB.TCPPARMS(TELNDBOF)
EZZ6045I TN3270B PROFILE PROCESSING COMPLETE FOR FILE
            TCPIPB.TCPPARMS(TELNDBOF)
EZZ6018I TN3270B PROFILE UPDATE COMPLETE FOR PORT    23
EZZ6038I TN3270B COMMAND OBEYFILE  COMPLETE
```

To avoid changing anything in the current profile (TELNDBON), TELNDBOF must be a copy of TELNDBON with these three statements replacing their equivalent statements in TELNDBON. (Both TELNDBON and TELNDBOF are based on the contents of TELNB30A).

Use VARY DEBUG OFF to turn off all debug options

As an alternative to using the OBEYFILE command and the DEBUF OFF statement, use the VARY TCPIP,,TELNET,DEBUG,OFF command. Using OBEYFILE only turns off debug for new connections. Any connections on the old profile continue to produce debug messages.

The VARY command can be issued to turn off DEBUG for *all* connections associated with *all* profiles, including the current profile. Summary messages for CONN DROP because of errors or time-outs will also be suppressed.

To turn on DEBUG again, issue a VARY TCPIP,,OBEYFILE command with the debug option specified in the TN3270 profile. Use DEBUG EXCEPTION to retain these messages.

The DEBUG OFF command is shown in Example 2-120.

Example 2-120 DEBUG OFF command used to turn off all debug options

```
V TCPIP,TN3270B,T,DEBUG,OFF
EZZ6038I TN3270B COMMAND DEBUG OFF COMPLETE
```

The options are listed again, after the DEBUG OFF command, as shown in Example 2-121.

Example 2-121 Display PROFILE to see DEBUG settings after DEBUG OFF command

```
D TCPIP,TN3270B,PROF,PORT=23,DET
EZZ6080I TN3270B PROFILE DISPLAY 132
...
 DIAGNOSTICS
   DEBUG CONN        VARYOFF
   DEBUG CONN        VARYOFF
   FULLDATATRACE
...
```

Use VARY ABENDTRAP to set abend traps

The **VARY TCPIP,,TELNET,ABENDTRAP,*module,rcode,instance*** command sets an abend trap based on unique Telnet return codes set in Telnet modules. You will normally use this command at the direction of IBM Support Center personnel. See *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details related to the use of this command.

Use NETSTAT CONN with APPLDATA

You can optionally display additional application connection data using the APPLDATA parameter on the NETSTAT CONN and NETSTAT ALLCONN commands. Example 2-122 contrasts the output of two NETSTAT CONN commands: one without the APPLDATA parameter 1, the other with the APPLDATA parameter 2.

The TN3270 server populates the APPLDATA field with connection data that is documented in Appendix “Application Data” found in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. The TN3270 appldata fields shown for the connection are the component ID, LU name, the SNA application name, connection mode, client type, security method, security level, and security cipher 3.

Example 2-122 NETSTAT CONN without and with the APPLDATA option

```
D TCPIP,TCPIPB,N,CONN                                    1
...
TN3270B  00000044 LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
TN3270B  00000067 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.8.21..23
  FOREIGN SOCKET: ::FFFF:10.1.100.221..1343
...

D TCPIP,TCPIPB,N,CONN,APPLDATA                           2
...
TN3270B  00000067 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.8.21..23
  FOREIGN SOCKET: ::FFFF:10.1.100.221..1343
  APPLICATION DATA:  EZBTNSRV SC31BB01 SC31TS03 ET B 3
...
```

You can filter the output of the NETSTAT CONN,APPLDATA command by adding the APPLD filter option and specifying the filter criteria. The APPLDATA field is a total of 40 bytes. By using an asterisk (*) in the filter criteria, you can wildcard on any part of the 40 bytes. Example 2-123 on page 138 shows a few filter criteria strings being used.

Example 2-123 NETSTAT CONN APPLDATA with APPLD filter

```
D TCPIP,TCPIPB,N,CONN,APPLDATA,APPLD=*TNSRV*
...
USER ID  CONN     STATE
TN3270B  00000067 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.8.21..23
  FOREIGN SOCKET: ::FFFF:10.1.100.221..1343
  APPLICATION DATA:  EZBTNSRV SC31BB01 SC31TS03 ET B
TN3270B  00000044 LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
  APPLICATION DATA:  EZBTNSRV LISTENER
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT

D TCPIP,TCPIPB,N,CONN,APPLDATA,APPLD=*SC31*
...
USER ID  CONN     STATE
TN3270B  00000067 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.8.21..23
  FOREIGN SOCKET: ::FFFF:10.1.100.221..1343
  APPLICATION DATA:  EZBTNSRV SC31BB01 SC31TS03 ET B
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

2.7.3 Use the MSG07 statement in the TN3270 profile

The MSG07 parameter is helpful when diagnosing problems. It allows TN3270 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 user without additional help.

Even for more difficult problems, MSG07 provides a good starting point. We suggest that MSG07 always be coded to send a notification to the user of connection failure issues, unless there are reasons not to send error messages to the client.

Use the MSG07 parameter statement to activate logon error message processing. Specifying this statement provides information to the client when a session attempts to the target application fails. If NOMSG07 is specified, the connection is dropped if a session initiation error occurs.

The MSG07 and NOMSG07 statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. They cannot be coded in the BEGINVTAM block.

MSG07 being used in the TELNETPARMS block is shown in Example 2-124.

Example 2-124 MSG07 used in TELNETPARMS block

```
TELNETPARMS
   . . .
   MSG07
   . . .
ENDTELNETPARMS
```

For details about its use and syntax, see the MSG07 statement description in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

2.7.4 Use SMF records to capture TN3270 connection activity

SMF records are written when a user establishes a session (SMF LOGN or Telnet SNA Session Initiation record, subtype 20) and when the session is ended (SMF LOGF or Telnet SNA Session Termination record, subtype 21). Optional SMF recording is controlled by using the SMFINIT and SMFTERM statements.

SMFINIT and SMFTERM can be coded in TELNETGLOBALS, TELNETPARMS, and PARMSGROUP. An example of SMFINIT/SMFTERM being used in the TELNETPARMS block is shown in Example 2-125. Example 2-125 turns off all 118 format records and uses standard 119 format records.

Example 2-125 SMFINIT/SMFTERM used in TELNETPARMS block

```
TELNETPARMS
   . . .
   SMFINIT 0 SMFINIT TYPE119
   SMFTERM 0 SMFTERM TYPE119
   . . .
ENDTELNETPARMS
```

For details about its use and syntax, see the SMFINIT/SMFTERM statement descriptions in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

2.7.5 Use trace data

When we need to look at the data being transmitted across the network, several types of trace tools available, including packet traces, data traces, and CTRACEs.

Use a packet trace to capture data buffers

Use the VARY TCPIP,,PKTTRACE command to trace data buffers between client and server. The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace. You will normally use this command at the direction of IBM Support Center personnel. See the following sources for details:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999
- *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis

Use a data trace to capture socket data

Use the VARY TCPIP,,DATTRACE command to trace socket data (transforms) into and out of the physical file structure (PFS). The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace. You will normally use this command at the direction of IBM Support Center personnel. See the following sources for details:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999
- *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis

Use the CTRACE system component to trace TN3270 processes

If a problem is not resolved using the preceding tools, IBM service will likely need a CTRACE with option Telnet. CTRACE, with only the Telnet option, gives complete information about the TN3270 processes. To debug almost any TN3270 problem, no other CTRACE option is needed. TN3270 running as its own procedure continues to use the SYSTCPIP component trace, but has fewer trace options to be specified.

A component trace SYS1.PARMLIB member is supplied for TN3270. The member name is CTIEZBTN. Trace setup for TN3270 running as its own procedure is the same as for the TCP/IP stack, except for having fewer trace options available.

For a complete discussion of the CTRACE options and trace setup, see *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782. Use *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis.

2.7.6 Tips for multiple TN3270E servers in a Parallel Sysplex environment

If multiple TN3270E servers are used (for example, multiple TCP/IP stacks in a sysplex environment) and if you do not use an LU name server to coordinate the LU name in a sysplex, ensure that each server uses unique LU names. Otherwise, the second server that uses the same LU name cannot establish a session. Either the OPEN ACB request will fail or the cross-domain session request will fail.

Note: Use an LU name server to centralize LU name allocation and to avoid duplicate LU name assignments among the group of Telnet servers, LU name requester. See 2.4, “Multiple TN3270E servers using LU name server and LU name requester” on page 95 for a detailed description about LU name server and LU name requester.

2.7.7 Tips for LU name server and LU name requester diagnosis

Use these tips when you diagnose LUNS and LUNR issues:

- Use the XCF GROUP display. Verify that all members have the same LU name server count, that there is an active LU name server, and that none of the LUNRs have the C status flag on. If an LU name requester has the C flag on, determine if there is a network connectivity problem and resolve it.
- Check the console for non-scrollable messages. If the profile pending message lingers, there probably is no LU name server. If the LU name server connection pending or rebuild pending messages linger, issue the XCF GROUP display to see which LUNRs are not communicating.
- If the PDMON flags C or R are on, or if you experience the loss of all connections with shared LU names assigned, the LU name requester might need more time to establish a connection to the LU name server after network failures take place. Increase the CONNECTTIMEOUT or RECOVERYTIMEOUT value or specify 0 to disable the functions.
- Issue the DISPLAY LUNS OBJECT command to verify the shared LU groups are defined at the LU name server. If not, verify you properly defined the LU name groups as shared. When issuing commands to view the LU name server LU table, make sure you specify LUNS. If you do not, you will get the Classic Telnet LU table.
- Issue the connection summary display to see if many connections have *LUNSREQ assigned. This is a special name used to indicate a request was sent to the LU name server but a reply has not been received yet. If you see several of these, check the connection between the LU name server and LU name requester.

2.8 Additional information sources for the TN3270E server

See the following sources for additional information about the TN3270E server:

- *3270 Data Stream Programmer's Reference*, GA23-0059
- RFC 1646: TN3270 Extensions for LUname and Printer Selection (July 1994)
- RFC 1647: TN3270 Enhancements (July 1994)
- RFC 2355: TN3270 Enhancements (June 1998) - Obsoletes RFC 1647
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS Communications Server: IP and SNA Codes*, SC31-8791
- *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999

3

File Transfer Protocol

File Transfer Protocol (FTP) allows you to move files between any computers that support the TCP/IP-standard FTP protocols—mainframes, midrange systems, PC servers, and desktop systems. FTP is the name of the application and the protocol that allows you to transfer files in a TCP/IP network. This chapter focuses on the FTP functions that are available in the z/OS Communications Server. The FTP protocol uses a client/server model and the z/OS Communications Server ships with both an FTP server (FTPD) and an FTP client.

This chapter discusses the following FTP topics.

Section	Topic
3.1, “Conceptual overview of FTP” on page 144	The basic concepts of FTP.
3.2, “Basic FTP without security” on page 148	Key characteristics of basic FTP and how to configure and verify a single server implementation.
3.3, “Multiple FTP servers in a sysplex” on page 173	How to implement multiple FTP servers in a Sysplex, one server per system image.
3.4, “FTP client using batch” on page 187	How to run an FTP client in batch.
3.5, “FTP client application program interface” on page 190	How to run FTP from a program.
3.6, “FTP access to UNIX named pipes” on page 192	Characteristics of named pipes and how to use in z/OS FTP environment.
3.7, “FTP large data set access” on page 199	A description of FTP large-volume access and how to use.
3.8, “Miscellaneous configuration settings of FTP” on page 201	Miscellaneous FTP functions not covered in detail in this book, such as REXX API, generic FTP server in a CINET environment, and the network management interface with SMF.
3.9, “Problem determination for FTP” on page 203	Problem determination techniques for FTP.
3.10, “Additional information sources for FTP” on page 203	References to additional reading sources for FTP.

3.1 Conceptual overview of FTP

As illustrated in Figure 3-1, FTP is one of the standard applications provided with the z/OS Communications Server. It executes under the z/OS UNIX environment, uses the Language Environment C-sockets API, and passes data through the UNIX Logical and Physical File Systems.

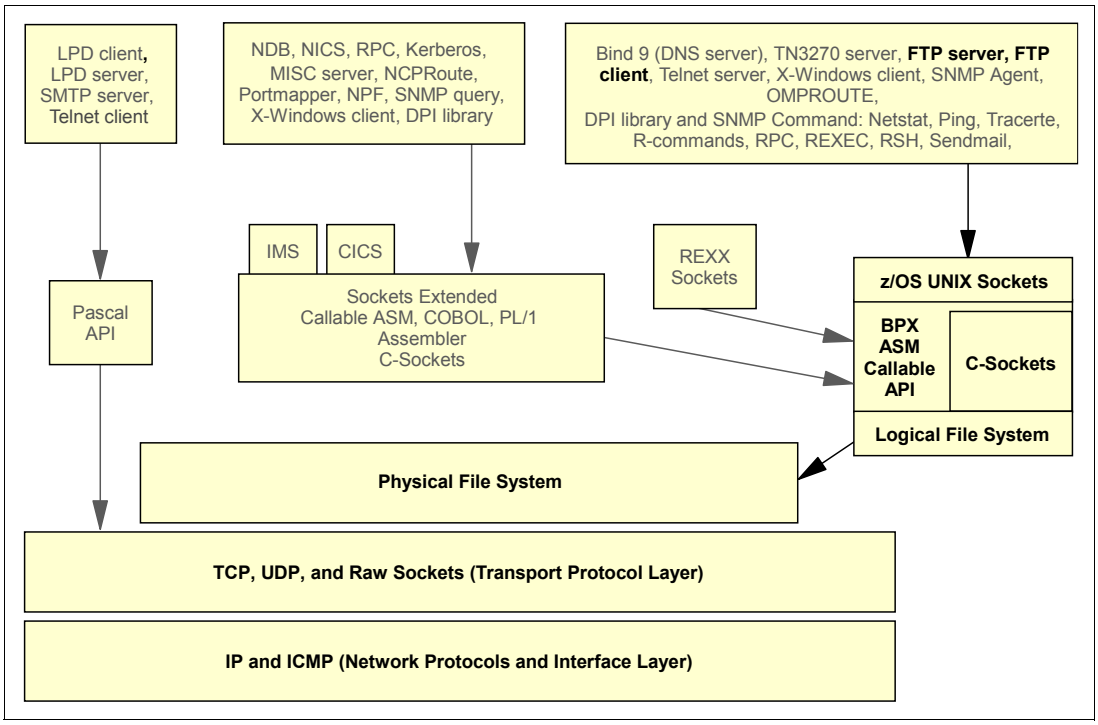


Figure 3-1 z/OS FTP client/server application services

This section includes the following topics:

- What is FTP
- How does FTP work
- How can FTP be used

3.1.1 What is FTP

FTP provides a fast and reliable method to transfer files in a TCP/IP network. FTP also allows for communication between platforms that have different character encodings and file structures by hiding such details from the user. With a few simple FTP commands, one can easily transfer a file from one platform to another regardless of whether the two platforms are similar.

There are two devices involved in an FTP session: the local host is the client, and a remote host is the server. Using the FTP command and subcommands, you can sequentially access multiple hosts without leaving the FTP environment. The local host or FTP client is the TCP/IP host that initiates the FTP session. The FTP server is the TCP/IP host to which the client's session is established. The server provides the responses to the client's commands and subcommands. z/OS Communications Server FTP includes translation facilities for ASCII/EBCDIC translation to support host file transfer to and from a variety of host platforms and operating systems.

3.1.2 How does FTP work

An FTP session is initiated when an FTP client connects to an FTP server using the TCP protocol. The FTP protocol requires the FTP server to use two TCP ports. One TCP port is the control connection over which information such as user ID and password is transmitted. All FTP commands, subcommands, and responses are exchanged over this connection. Well-known port 21 is used as the default for the control connection port on the FTP server. The other TCP port is the data connection, which is used for transferring the contents of files based on the FTP client's requests. The output of the **`ls`** or **`dir`** FTP subcommands is also sent over the data connection. Well-known port 20 is the default for the data connection port on the FTP server. See *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, for details about FTP usage, commands, and subcommands.

During an FTP session it is important to keep track of which machine is the client and which is the server, because this determines whether you use a **`get`** command or a **`put`** command to move files. The **`get`** command is always used to copy files from the server to the client and the **`put`** command is used to copy files from the client to the server.

Configuration files

The FTP server and client can each have its own optional FTP.DATA configuration data set. The z/OS Communications Server provides a sample FTP.DATA for the FTP server in SEZAINST(FTPSDATA) and a sample for the FTP client in SEZAINST(FTCDATA). *z/OS Communications Server: IP Configuration Reference*, SC31-8776, covers the configuration statements in detail and indicates which statements are appropriate for the server or the client.

FTP server job name

The FTP server forks once at startup, and forks again each time a new connection is accepted. The job name of the initial address space is based on the started procedure. The server's listener thread (first forked thread) always adopts the started task's name (proc name) suffixed with the number 1. For example, if the proc name is FTPD, then the listener's name is always FTPD1. For all subsequent forks, a new job name is chosen in one of three ways:

- If the proc name is already eight characters long, then the job name remains that same eight character name, and does not change across forks.
- Assigned by _BPX_JOBNAME environment variable
- Assigned by z/OS UNIX

_BPX_JOBNAME

If the original job name is less than eight characters, and *_BPX_JOBNAME* is made available to the FTPD server at startup, all forked threads, after the listener's, result in the job name specified by *_BPX_JOBNAME*.

Assigned by z/OS UNIX

If the original job name is less than eight characters, and *_BPX_JOBNAME* is not specified, then z/OS UNIX assigns forked job names by appending a number to the job name. For example, if a started procedure named FTPD is used to start the FTP server, then the job name of the listener is FTPD1. For each subsequent fork (which is created in behalf of a new client connection), the job name becomes FTPD*x,* where *x* is a number between two and nine (FTPD2 - FTPD9).

z/OS UNIX uses RACF's *BPX.DAEMON* FACILITY class profile to manage the UNIX security context of threads that are forked by certain servers. These servers change the security environment of the process in which they execute in order that the client threads execute

under the security permissions set up for the client's user ID. 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 logged-in user, instead of the server's user ID. The RACF FACILITY class resource *BPX.DAEMON* is used by the servers for changing the security context during this process.

3.1.3 How can FTP be used

An FTP client program and a separate FTP server program are provided with z/OS Communications Sever.

The relationship between the FTP client and FTP server is shown in Figure 3-2.

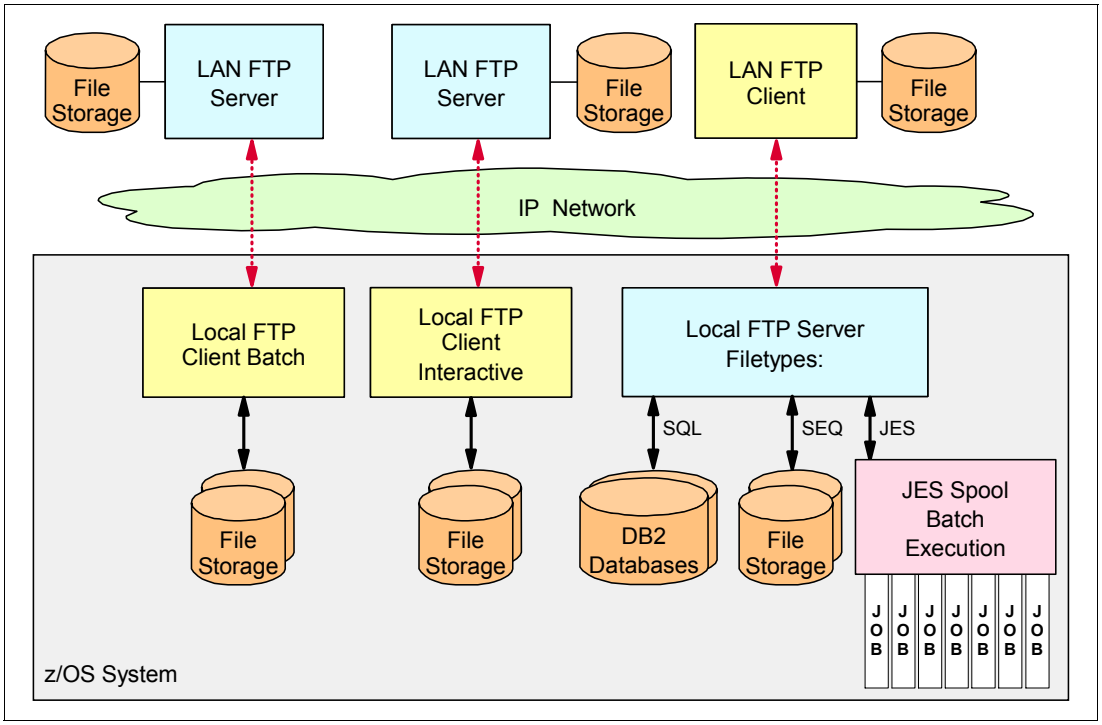


Figure 3-2 FTP client/server relationships

Features of FTP

FTP in the z/OS Communications Server includes the following features:

- Access to traditional MVS data sets (including PDS and PDS/E data sets)
- Access to files in the z/OS UNIX file system
- Support for MVS batch jobs using the JES file type
- Support for SQL queries using the SQL file type
- Translation tables supporting many different languages, including single-byte character sets (SBCS), double-byte character sets (DBCS), and unicode character set 2 (UCS-2)
- Secure communications using Transport Layer Security (TLS) or Kerberos
- Secure communications using Application Transparent TLS (AT-TLS)
- Access control through support for user exits
- Support for user login with password or password phrases

- Support for anonymous login
- A customizable welcome banner, login message, and directory readme files
- Access to traditional MVS data sets (including PDS and PDS/E data sets)
- Invoking of client from TSO, from the z/OS UNIX shell, as a batch job, or by an application programming interface (API)
- Support for FTP proxy connections using a SOCKS server
- Controls in FTP.DATA for server and client operations
- Support for UNICODE TP reporting of UTF-8, UTF-16, UTF-16LE, and UTF-16BE encoding

The client and the server can both be executed with or without encryption security. Both can support SOCKS proxy protocols to accommodate file transfers within a firewall environment. A single generic FTP server can be implemented on a single LPAR that has multiple stacks. The single server can listen on each stack. Load balancing and availability requirements across FTP servers within a sysplex can be implemented. Multiple FTP server instances can be implemented in a multiple stack (CINET) environment. The FTP client can run in batch, in a TSO environment, or under program control using one of the available API interfaces.

Setting up

For users interested in setting up basic FTP without security to get a simple FTP server up and running, use a configuration similar to the one described in 3.2, “Basic FTP without security” on page 148. You can then later add security definitions to the basic configuration.

For users interested in secure FTP, refer to using the native TLS configuration or using a stack-wide security solution such as Application Transparent Transport Layer Security (AT-TLS) or IPSec. In addition to native TLS support, the FTP server and client can use AT-TLS to manage TLS security. TLS managed by AT-TLS (TLSMECHANISM ATTLS) supports more security functions than TLS managed natively by the FTP (TLSMECHANISM FTP). Be aware of these AT-TLS capabilities and requirements when planning the preferred AT-TLS support for FTP:

- Specify the label of the certificate to be used for authentication instead of using the default
- Support SSL session key refresh
- Support SSL sysplex session ID caching
- Trace decrypted SSL data for FTP in a data trace
- Receive more detailed diagnostic messages in syslogd
- There are no restrictions for the FTP ATTLS function.
- Policy Agent must be active for FTP ATTLS to work.
- TLS security defined natively to FTP will continue to be available in addition to AT-TLS.

You can use the VERIFYUSER statement to indicate whether the FTP server should verify that every user ID used to log in to FTP is granted access to the server’s port profile in the SERVAUTH class:

- The FTP server port profile is the same profile that is checked for TLS secured sessions when SECURE_LOGIN VERIFY_USER is coded in FTP.DATA. See *z/OS Communications Server: IP User’s Guide and Commands*, SC31-8780.
- When sessions are secured with TLS and VERIFYUSER TRUE is coded in FTP.DATA, the server verifies the user access to the FTP server port profile regardless of the SECURE_LOGIN value.

However, FTP can verify new users against the AT-TLS SAF resource without requiring the use of AT-TLS (VERIFYUSER function).

All TLS and AT-TLS security-related scenarios are discussed in *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

For users who are interested in load balancing FTP across multiple LPARs, we recommend the high availability scenarios, as discussed in *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998.

We describe the following scenarios in this chapter:

- Basic FTP without security
- Multiple FTP servers in a sysplex
- FTP client using batch
- FTP client application program interface
- Miscellaneous configuration settings of FTP

3.2 Basic FTP without security

In this section we discuss FTP configured with common configuration options but without any data security beyond user IDs and passwords. In practical terms, we discuss the transmission of FTP data without any encryption. This is the easiest and quickest way to get a basic FTP server configuration implemented.

We discuss the following topics:

- Description of basic FTP without security
- Planning for the basic FTP environment without security
- Configuration of basic FTP without security
- Activation and verification for basic FTP without security

3.2.1 Description of basic FTP without security

This section provides information about items to consider when running basic FTP without security.

Dependencies of basic FTP

Implementing FTP without security requires minor changes to the TCP/IP configuration, some definitions in the SERVAUTH SAF class, and an FTP.DATA data set. You should start syslog prior to starting FTP to retain messages from the FTP server.

Advantages of basic FTP

FTP running without security requires less overhead than FTP running with security.

Alternatives for basic FTP

Without using z/OS Communications Server security capabilities, you might need to implement necessary security elsewhere (in your firewalls or routers, for example) or you can elect to implement some level of IPSec.

3.2.2 Planning for the basic FTP environment without security

Each operating system has unique requirements for allocating files or data sets in its file system. These requirements differ so widely between operating systems that it has been impossible to develop a single protocol that embraces all requirements for all operating systems. To cover all requirements, the FTP protocol implements a SITE command which enables an FTP client to send site-specific parameters to the FTP server over the control connection.

Option to change server reply code from 250 to 226

This option can improve information management and log analysis. It enables you to configure FTP server to reply with 226 instead of 250 after a successful file transfer. It provides more flexibility when obtaining access through firewalls.

Generally, reply code 226 or 250 is used after a successful file transfer, after LIST commands, and after NLST commands. Reply code 250 (not 226) is used for a broader class of FTP commands, such as RNTO, DELE, MKD, RMD, CWD.

The FTP server's current implementation sends a 250 reply, even though it has already closed the data connection. This should be changed to 226 to comply with the RFC standard. This change is being implemented using an FTP.DATA parameter to allow local control in case there is a local dependency on the 250 reply.

Use the `REPLY226` statement to direct the FTP server to reply to the FTP client with reply code 226 instead of reply code 250 to command sequences described in RFC-959 that allow the server to choose between reply 226 and reply code 250.

The options for this statement are as follows:

- `FALSE`

  Directs the server to reply to the client with code 250 after successful file transfer, and after other FTP commands where the server is allowed to choose between reply code 250 and reply code 226. This is the default.

- `TRUE`

  Directs the server to reply to the client with reply 226 instead of reply code 250 after successful file transfer, and after other FTP commands where the server is allowed to choose between reply code 250 and reply code 226.

Restriction: A server is not always permitted to select reply 226 instead of reply 250. The REPLY226 setting does not override RFC-959 in these cases.

For example, RFC-959 stipulates the server must reply with reply 250 to RMD (remove directory); the REPLY226 setting does not affect the reply code for RMD commands.

Parameter for FTP client to specify FTP.DATA

An option at the client invocation level enables the FTP client to specify a data set, UNIX file, or ddname to override the FTP.DATA search order. This option is implemented using an FTP.DATA parameter to allow customer control over its usage.

The addition of the **-f** parameter changes the FTP.DATA search order, as listed in Table 3-1.

Table 3-1 Option to set the FTP.DATA file through the -f parameter

TSO environment	z/OS UNIX System Service shell
0. -f parameter 1. SYSFTPD DD statement 2. tso_prefix.FTP.DATA 3. userid.FTP.DATA 4. /etc/ftp.data 5. SYS1.TCPPARMS(FTPDATA) data set 6. tcpip_hlq.FTP.DATA	0. -f parameter 1. $HOME/ftp.data 2. userid.FTP.DATA 3. /etc/ftp.data 4. SYS1.TCPPARMS(FTPDATA) 5. tcpip_hlq.FTP.DATA

Note: It is important to specify an existing ftp.data file with the -f parameter. If the specified file is not found, then the standard ftp.data search order is performed. No error message is issued.

Platform-specific features using SITE/LOCSITE commands

When a user on your z/OS system starts the FTP client, a set of default local SITE parameters is in effect. The default values can be specified in the FTP.DATA data set. If an FTP.DATA data set cannot be found, a set of hard coded values is used. The user can change these parameters during the FTP session by using the LOCSITE command.

For details about the SITE, LOCSITE, and FTP.DATA client statement parameters, consult *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Data set attributes

When an FTP client issues a **`put`** to transfer a file to the z/OS FTP server, the FTP server needs specific parameters to allocate a data set. These parameters include record format (RECFM), record length (LRECL), unit type (UNIT), and blocksize (BLKSIZE), plus many others, depending on the specific operation requested. The FTP server has a set of default values for all the parameters it might need. The client can change many of these values for the current FTP session using the SITE command.

If you use the z/OS FTP client function and you retrieve a file from an FTP server somewhere in your IP network, the FTP client also needs a set of parameters similar to those of the z/OS FTP server to allocate a data set in MVS. Again, a set of default values exists for the z/OS FTP client, but a user can change these using the LOCSITE command.

You do not necessarily need to specify all the allocation attributes of an MVS data set; you can instead use the Storage Management System (SMS) of IBM Data Facility Systems Managed Storage. You have in both the SITE and the LOCSITE command an option to specify values for the three main SMS parameters: dataclass, mgmtclass, and storclass. These SMS options are as follows:

- Data class (`site/locsite dataclass=`), which is a collection of data set allocation attributes, for example, space requirements, record format, data set type, or retention period.
- Management class (`site mgmtclass=`), which is a collection of management attributes, for example, migration rules, backup frequency, or rules for release of unused space.
- Storage class (`site storclass=`), which is a collection of service attributes, for example, availability requirements and requested storage subsystem response time.

Consult your storage administrator for a list of available SMS parameters in your installation.

Directory mode or data set mode

The directory mode or data set mode specifies how the output from a directory command submitted to the z/OS FTP server should look. Working with FTP employs the notion of a directory and a hierarchy of directories. When MVS is the FTP server, the client still uses the directory notion and the server must transform this notion into the traditional MVS file system structure. The client can switch between the two modes by using the SITE command specifying either DIRECTORYMODE or DATASETMODE.

DATASETMODE is the normal MVS method of displaying MVS data set names. An example of this output is shown in Example 3-1.

Example 3-1 Output of dir command in DATASETMODE

```
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit  Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
WTLSTG 3380K  05/13/96  1   10  FB      80  6160  PO  DB2.CNTL
WTLSTG 3380K  05/13/96  1    6  VB    4092  4096  PS  DB2.OUTPUT
WTLSTG 3380K  05/13/96  1    2  FB      80  3120  PO  ESA4.ISPPROF
WTLSTG 3380K  05/13/96  1    9  FB      80  6160  PO  ISPF.CLIST
WTLSTG 3380K  05/13/96  1   10  FB      80  6160  PO  ISPF.ISPPLIB
WTLSTG 3380K  05/13/96  1    1  FB      80  6160  PO  ISPF.TEST.CLIST
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  ISPF.TEST.WORKDSN
WTLSTG 3380K  05/13/96  1    2  FB      80  3120  PO  ISPFESA.ISPPROF
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  PRINT
WTLSTG 3380K  05/13/96  1    8  VA     125   129  PS  SPFLOG1.LIST
WTLSTG 3380K  05/13/96  1    1  FB      80   800  PS  SPFTEMP1.CNTL
250 List completed successfully.
808 bytes received in 1.3 seconds (0 Kbytes/s)
```

However, we might want output that more closely resembles the UNIX style. DIRECTORYMODE uses the value of your TSOPREFIX setting in RACF as your default directory. If you do not maintain TSO logon information in RACF, your user ID will be used as your default directory.

Each qualifier level in the data set name is considered a directory. A directory can contain data sets or subdirectories. A partitioned data set is considered a directory, and the individual members as files in that directory. You can step down the hierarchy by using CD commands to name the next low-level qualifier you want to view. You can step up to the root by using CD commands.

An example of DIRECTORYMODE is shown in Example 3-2.

Example 3-2 Output of dir command in DIRECTORYMODE

```
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit  Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
Pseudo Directory                                    DB2
Pseudo Directory                                    ESA4
Pseudo Directory                                    ISPF
Pseudo Directory                                    ISPFESA
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  PRINT
Pseudo Directory                                    SPFLOG1
Pseudo Directory                                    SPFTEMP1
250 List completed successfully.
493 bytes received in 0.84 seconds (0 Kbytes/s)
```

Data type, structure, and mode

At first glance, it might seem to be a trivial matter to transfer files between different computer systems, but when you take a closer look you soon discover a range of issues created by the diversity of computer architectures in a typical IP network. Some operating systems use 7-bit ASCII to represent character data. Others use 8-bit ASCII or EBCDIC, to mention the most obvious. Some operating systems organize files into records; others treat files as continuous streams of data, possibly without any encoded notion of record boundaries. (In this case, it is up to the program reading or writing the data to impose a structure onto the data stream.)

The FTP protocol can help deal with these issues, but you must select the proper options to let FTP transfer a file in such a way that it is usable on the receiving system.

FTP always transfers data in 8-bit bytes, the *transfer size*. If the sending or receiving system uses another byte length, it is up to the FTP client and the FTP server to implement the proper conversion between local byte sizes and the FTP transfer size. When FTP transfers ASCII data, it always transfers it in 8-bit bytes, where the bits are used to encode the ASCII character according to a specific ASCII standard, which is called NVT-ASCII (Network Virtual Terminal ASCII as defined in the TELNET protocol). This implies that when you transfer ASCII type data between two ASCII hosts, a translation from the local ASCII representation to NVT-ASCII for transmission and back to the receiving hosts local ASCII representation always takes place.

When MVS is involved in an ASCII type transfer MVS translates the received NVT-ASCII into EBCDIC and translates data to be transmitted from EBCDIC into NVT-ASCII.

When you request an FTP file transfer, you can characterize the transfer by means of three attributes: type, structure, and mode.

Data type

Data type, also known as transfer type or representation type, indicates how the bits of data should be interpreted by the receiver. There are three values: ASCII, EBCDIC, and IMAGE.

ASCII When you set the data type to ASCII, the receiver knows that the data is character data and that each line of data is terminated using a control sequence of Carriage Control plus Line Feed (CRLF), which in ASCII is X'0D0A'. If MVS is the receiving side, data is translated from NVT-ASCII to EBCDIC and the CRLF sequences are removed from the data stream and substituted by traditional MVS record boundaries according to the current settings of the SITE/LOCSITE parameters: RECFM and LRECL. If RECFM is fixed, the data records are padded with extra spaces to fill up a record. If MVS is the sending side, the data is translated from EBCDIC into NVT-ASCII and, based on existing record boundaries, CRLF sequences are constructed and inserted into the ASCII data stream. A data type of ASCII is the default data type in all FTP implementations.

EBCDIC A data type of EBCDIC means that the data transferred is EBCDIC data. In such a case, no translation to NVT-ASCII or from NVT-ASCII takes place in MVS. The 8-bit EBCDIC bytes are transferred as they are. If you transfer text data between two EBCDIC systems, a data type of EBCDIC is the most efficient way to transfer the data. Most ASCII hosts will reject a data transfer where you specify a data type of EBCDIC. Some will treat it as an ASCII transfer, but the point where the translation takes place is at their end of the FTP connection, and not in MVS.

IMAGE A data type of IMAGE means that the data will be transmitted as contiguous bits packed into the 8-bit FTP transfer byte size. No translation takes place, either at the sending or at the receiving side. You normally use this data type for

binary data, such as program files. If you transfer text data between two similar ASCII hosts, it will often be more efficient to use an IMAGE data type instead of an ASCII data type. As the two systems use exactly the same ASCII representation, there is no need to impose the overhead of translating to and from NVT-ASCII.

Both ASCII and EBCDIC data types have a second attribute, *format control*, which can have three values:

Non-print The text data does not include any vertical format control characters. This format control is the only one you will find in the MVS FTP implementation. When you set data type to ASCII, the format control defaults to non-print.

TELNET The text data includes TELNET control characters. This is not commonly used.

CC The text data includes Carriage Control (ASA) control characters, according to the FORTRAN implementation.

Data structure

Structure refers to how the data is stored by the receiver. The three possible values are file, record, and page:

File The file has no internal structure and is considered to be a continuous sequence of bytes. File structure can be used with all transfer modes and data types, and is the most widely implemented.

Record The file is made up of sequential records. An example is ASCII records with CRLF characters. All data types are generally supported for record structure. In CS for z/OS IP, the explicit use of record structure is only supported with stream mode transfers. When record structure is explicitly used, each record is terminated by an end-of-record (EOR) sequence, which is X'FF01'. End-Of-File (EOF) is indicated by X'FF02'. If the data contains X'FF' bytes, each X'FF' byte is transmitted as two bytes, X'FFFF', to preserve the original value. In CS for z/OS IP both the FTP server and the FTP client can support this record structure.

Page A third structure value is page structure. It is not used in conjunction with MVS, and CS for z/OS IP does not support it, either in the FTP client or in the FTP server.

Transfer mode

Transfer mode refers to the organization of the data as it is transmitted. The three possible values are stream, block, and compress.

Stream Data is transmitted as a stream of bytes. The data is passed with little or no extra processing. With stream mode, the data type is used to determine if any processing at all should be applied to the data, such as translation of text data or CRLF processing. There is no restriction on data type or data structure. If record structure is used, the end of file is indicated through the EOF control sequence (`X'FF02'`). If file structure is used, the end of the file is indicated when the sending host closes the data connection. Stream mode is the default transfer mode, and the most commonly used.

Block In block mode data is sent as a series of data blocks. Each block is preceded by a header. The header contains a count field of the number of bytes in the block (excluding the header itself) and a descriptor code, which defines block attributes (last block in the file, last block in the record or restart marker). The FTP protocols do not impose any restrictions on either data type or structure used with block mode transfers. The actual FTP implementations do, however, impose restrictions of various kinds. In CS for z/OS IP, for example, block mode

transfer is only supported with a data type of EBCDIC. You can use block mode when you transfer files between z/OS hosts. A file transferred between two MVS systems in block mode preserves its record structure unchanged, including files with variable length records

Compress Data is transmitted in a compressed format. The compression algorithm is rather simple. It includes the ability to send replicated bytes in a two-byte sequence (maximum 128 replications), and to send a filler byte in a one-byte sequence (maximum 64 filler bytes). A filler byte is defined as *space* for ASCII or EBCDIC data types and as binary zero for a data type of image. In CS for z/OS IP compressed mode requires a data type of EBCDIC.

Table 3-2 provides an overview of the supported combinations of data type, structure, and mode. Table 3-2 also provides cross-references between mode, type, and structure. The various options are also discussed in greater detail in the following sections.

Table 3-2 Cross-references between mode, type, and structure

	Data type			**Data structure**	
Transfer modes	**ASCII**	**EBCDIC**	**Image**	**File**	**Record**
STREAM	Yes	Yes	Yes	Yes	Yes
BLOCK	No	Yes	No	Yes	No
COMPRESSED	No	Yes	No	Yes	No

When you select among the options listed, consider the purpose of your file transfer:

- Are you going to transfer a file to a host, where the file will be processed by programs on that host? In that case, you must select options that will result in a file that can be used by the target host. If the data is text, the originating host uses EBCDIC, and the target host uses ASCII, you must use an ASCII data type and a stream transfer mode.
- Are you going to transfer a file to another host for intermediate storage only and later retrieve it again on the original host? In this case it is important that you can invert the process, so the file you will end up with back on your original host is exactly like the file you started with. If it is text data, you might not need to translate between EBCDIC and ASCII, and you can use the BINARY data type instead.

FTP STATUS and LOCSTAT subcommands

The FTP client subcommands STATUS and LOCSTAT have the ability to display status selectively. This is accomplished with the use of a parameter. The STAT subcommand displays the status of parameters in the *server*; the LOCSTAT subcommand displays the output of *client* status information.

Restriction: Only one argument can be used at a time with the STAT and LOCSTAT subcommands. An example of the **`stat`** and **`locstat`** commands is shown in Example 3-3.

Example 3-3 Use of STAT and LOCSTAT commands

```
ftp> quote stat (autor                1
211-Automatic recall of migrated data sets.
211 *** end of status ***
ftp>

EZA1460I Command:
EZA1736I   STAT (INACTIVETIME         2
EZA1701I >>> XSTA (INACTIVETIME
211-Inactivity timer is set to 300
211 *** end of status ***

EZA1460I Command:
EZA1736I   LOCSTAT DCONNTIME          3
EZA2811I DCONNTIME is 120
```

In this example, the numbers correspond to the following information:

1. The STAT command issued to a z/OS server from a remote non-z/OS client that does not know what the STAT command is must send the command to the server using the QUOTE command.
2. A local z/OS client sends the STAT command to a z/OS server.
3. The LOCSTAT command being sent to the local z/OS client.

Transferring MVS data sets to stream-oriented file systems

If you want to use a stream-oriented file system as intermediate storage for a record-oriented MVS file then you might have a problem, depending on the record format of the MVS data set you want to store and the data type you use. For ASCII data types, record boundaries do not impose problems. The record boundaries are preserved by means of CRLF (Carriage Return, Line Feed - X'0D0A') for DOS-based systems[1] or just LF (Line Feed - X'0A') for UNIX-based systems. If such a data set is transferred from MVS to, for example, UNIX and back to MVS again, the CRLF or LF is used to rebuild the record boundaries and the data set will be identical to the one you originally had on MVS. This is true for both fixed length and variable length record data sets.

For BINARY or IMAGE transfer from MVS to a stream-oriented file system, the situation is slightly more complicated. When the records of an MVS data set are stored in a stream-oriented file system, the records are stored one after the other as one long stream of bytes, without any notion of the record boundaries from the MVS system.

If the original data set was a fixed length record data set, you can reconstruct this data set if you transfer the file back to MVS using the same logical record length as the original data set. The long stream of bytes is chopped into records of exactly the length you specify, thereby reconstructing the same record boundaries as you had in the original data set.

If the original data set was a variable length record data set or a data set with undefined record format, you cannot use the above technique because no knowledge length of each record in the original data set has been retained. There are two ways in which you can move a variable record length file out of MVS and back into MVS again, preserving the record boundaries:

- Use the RDW option of the z/OS FTP client or z/OS FTP server.
- Use the record structure option.

[1] Including all the descendants of DOS.

Note: We strongly recommend that you use the record structure option. This is one of the standard file structures defined in RFC959. Both the FTP server and FTP client support it.

Using the RDW option

If you use the FTP client or the FTP server to transfer a variable length record data set from MVS to a stream-oriented file system and you use the RDW option, the file stored on the stream-oriented file system will include the record descriptor words (RDWs) of each record in the original data set.

If the purpose of including the RDWs was to let an application program on the remote host work with the information in the file including the RDWs, you have accomplished what you wanted, but there might be situations in which you might want to get such a file back into MVS again. Unfortunately, the code in CS for z/OS IP that stores the data set on MVS DASD does not use the preserved RDWs to reconstruct record boundaries. Instead, the DCB information given in either the SITE or the LOCSITE command is used. You can implement a solution to this problem using the following method:

1. Use the z/OS FTP client or the z/OS FTP server to transfer a RECFM=V or VB data set to Linux, for example, using the BINARY, STREAM, and RDW option. This will give you the file on Linux with imbedded RDWs.
2. Transfer the file back to MVS using the BINARY, STREAM, and MVS SITE parameters of RECFM=U and BLKSIZE=some high value.
3. Create a program that, based on the imbedded RDWs, reconstructs the original record structure.
4. Be careful using the RDW option with ASCII transfers. Transferring the file out of MVS will work without problems, but if you later want to transfer the file back into MVS, the ASCII-to-EBCDIC translation will also translate the RDWs, which might have unexpected results.

Using the FTP record structure option

When you connect an FTP client to the z/OS FTP server, you can use the record structure option to transfer a variable length record data set from MVS to a stream-oriented file system without the need to deal with RDWs.

Retrieving a recfm=vb data set from the z/OS FTP server to a non-z/OS client

Complete the following steps:

1. Connect your FTP client to the z/OS FTP server and set transfer mode to binary.
2. Issue a **`dir`** command and make a note of the record format (which should be VB), record length, and block size. You need this information later if you want to return this data set back to the z/OS FTP server.
3. Issue a **`quote stru r`** command. This command is not interpreted by the FTP client, but sent directly to the z/OS FTP server. The effect of this command is that the z/OS FTP server sends data to the FTP client with imbedded end-of-record sequences.
4. Issue a **`get`** command to copy your variable record length file from z/OS to the client. Because the **`structure`** command was sent in quotation marks, the client does not know about it and will receive and store the file as a binary stream of bytes, including the imbedded EOR sequences. When you want to copy the file back into MVS again, connect your FTP client to the z/OS FTP server, set transfer mode to binary, and send the **`quote`** command with a **`structure`** operand telling the z/OS FTP server to expect records with imbedded EORs.

Sending a recfm=vb data set from a non-z/OS client to a z/OS FTP server

Complete the following steps:

1. Connect your FTP client to the z/OS FTP server and set transfer mode to binary.
2. Issue a **quote stru r** command. This command is not interpreted by the FTP client, but is sent directly to the z/OS FTP server. The effect of this command is that the z/OS FTP server receives data from the FTP client and recognizes the embedded end-of-record sequences.
3. Depending on your default SITE parameters, you might need to send a SITE command with record format and record length information to the z/OS FTP server before you issue your **put** for the file.
4. Issue a **put** command. Because the FTP client does not know about the record structure, it transfers the file as a stream of bytes. The file still has the imbedded EOR sequences that are interpreted by the z/OS FTP server to rebuild the original record boundaries.

Although this technique can be used to transfer the VB data set in binary mode, it is still difficult to use the contents of a file at the remote system, because the file received contains imbedded EOR sequences. Any manipulation of the file on the remote server must be careful to preserve the format of the file.

The FTP client included in the z/OS Communications Server can support the record structure option. Therefore, you can transfer any VB files using structure mode easily between MVS systems.

Transferring a recfm=vb data set between z/OS systems

Complete the following steps:

1. Connect the z/OS FTP client to the z/OS FTP server and set transfer mode to binary.
2. Issue a **stru r** command. Because both FTP server and client can recognize the **stru r** command, you do not need to add the **quote** command in front for a z/OS-to-z/OS transfer. Both FTP server and client will recognize the file structure as record.
3. Issue a **put** or **get** command.

FTP UTF-8 data transfer and storage

UNICODE provides a unique number for every character, no matter what platform, program or language you are using. The same code page (UNICODE) is supported regardless of which operating system it runs on (z/OS, Windows, Linux, AIX, and so on). When storing UNICODE files you have the choice of retaining, discarding, or creating a byte order mask. This facilitates upload of UNICODE files from workstations to mainframe print solutions that support UNICODE.

Within UNICODE, there are multiple encoding methods. This function addresses only UTF-8 encoding. This implementation of UNICODE is a step forward in moving z/OS to fuller Unicode enablement.

Considerations for using UTF-8 support

For our usage we needed to observe the following points:

- This implementation will be used to transfer UTF8-encoded data from other platforms to z/OS and to maintain its characteristics solely for the purpose of printing UTF-8 encoded files on z/OS system printers.
- Errors in the UTF-8 stream cannot be detected by FTP when you specify MBDATACONN=(UTF-8,UTF-8) to transfer UNICODE files.

- This implementation enables the use of z/OS systems as data repositories for UNICODE data, and the use of a high speed printing system as a printing solution for UNICODE data.
- The FTP client must provide support for transferring files using UNICODE when you use a **get** to store the files in your client.
- z/OS FTP allows users to specify whether files stored as UTF-8 are stored with a byte order mask.
- The purpose of a byte order mask is to indicate whether a UNICODE data stream is encoded with *Little Endian* or *Big Endian*, which refers to the byte order in which multiple numbers are stored.
  - Little Endian means that the low-order byte of the number is stored in memory at the lowest address, and the high order byte at the highest address.
  - Big Endian means that the high-order byte of the number is stored in memory at the lowest address, and the low-order byte at the highest address.
- For UTF-8, the value of the byte order mask is EF BB EF. This sequence can appear anywhere in the file, but it is considered a byte order mask only when it appears in the first character position of the file.

Configuration of UTF-8 support

We used a Windows system to generate the UNICODE file and transfer it to the z/OS FTP Server, which supported the UTF-8 encoding data transfer. One might think that we could transfer UTF-8 simply by setting TYPE to IMAGE (binary) before sending the files. However, a binary transfer does not translate newline markers to EOL when sending, and the EOLs are not translated to newline markers when receiving.

The use of FTP UTF-8 is invoked by using the following statements and subcommands:

```
ENCODING=MBCS
TYPE=ASCII
MBDATACONN=(UTF-8,UTF-8)
UNICODEFILESYSTEMBOM
MBREQUIRELASTEOL
```

These statements and subcommands can be coded in the z/OS FTP server FTP.DATA file, the z/OS FTP client FTP.DATA file, or issued using the SITE command for the server or the LOCSITE subcommand for the client. Type, ENCODING and MBDATACONN are existing FTP configuration options and subcommands.

UNICODEFILESYSTEMBOM is used to specify whether to add a Byte Order Mark (BOM) to a file stored in the local file system when the file system code page is UNICODE. The options are as follows:

- ASIS

  If a Byte Order Mark is present in a UNICODE file that is received from the network store the file with a Byte Order Mark. If a Byte Order Mark is not present, store the file without a Byte Order Mark. This option is the default.

- ALWAYS

  Always include a Byte Order Mark when storing the file. If the file is received without a Byte Order Mark, insert a Byte Order Mark into the file.

- NEVER

  Never include a Byte Order Mark when storing a UNICODE file. If the file is received with a Byte Order Mark, discard it before storing the file.

When appending to a nonexistent file, the FTP server respects the UNICODEFILESYSTEMBOM setting. However, when appending to an existing file, the FTP server always strips a leading Byte Order Mark from the incoming file. This prevents a superfluous BOM from being inserted in the midst of a server file.

MBREQUIRELASTEOL is used to specify whether FTP will require the last record of incoming multibyte files to end with the FTP standard EOL sequence. This setting applies when the server is receiving a multibyte file from the client, and when the client is receiving a multibyte file from the server. If you set MBREQUIRELASTEOL to FALSE, and you have coded CHKCONFIDENCE TRUE in FTP.DATA, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record will be high. If you set MBREQUIRELASTEOL to TRUE, and you have coded CHKCONFIDENCE TRUE in FTP.DATA, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record is low.

MBDATACONN is used to define the conversions between file system code page and the network transfer code page during data transfer. It affects the conversion of double-byte character set (DBCS) and multibyte character set (MBCS) data and is used when the ENCODING MBCS statement is coded. The SITE and LOCSITE subcommands are also available to set this keyword. The options (UTF-8,UTF-8) are used to support UNICODE encoding. Example 3-4 shows the use of these parameters.

Example 3-4 FTP.DATA with parameters and options for UNICODE.

```
Encoding        MBCS       ;
MBdataconn  (UTF-8,UTF-8)  ;
Unicodefilesystembom ASIS  ;
MBrequirelastEOL TRUE      ;
```

To create UTF-8 file on our workstations we used the Windows notepad to edit a file and save it with the appropriate encoding option, as shown in Figure 3-3.

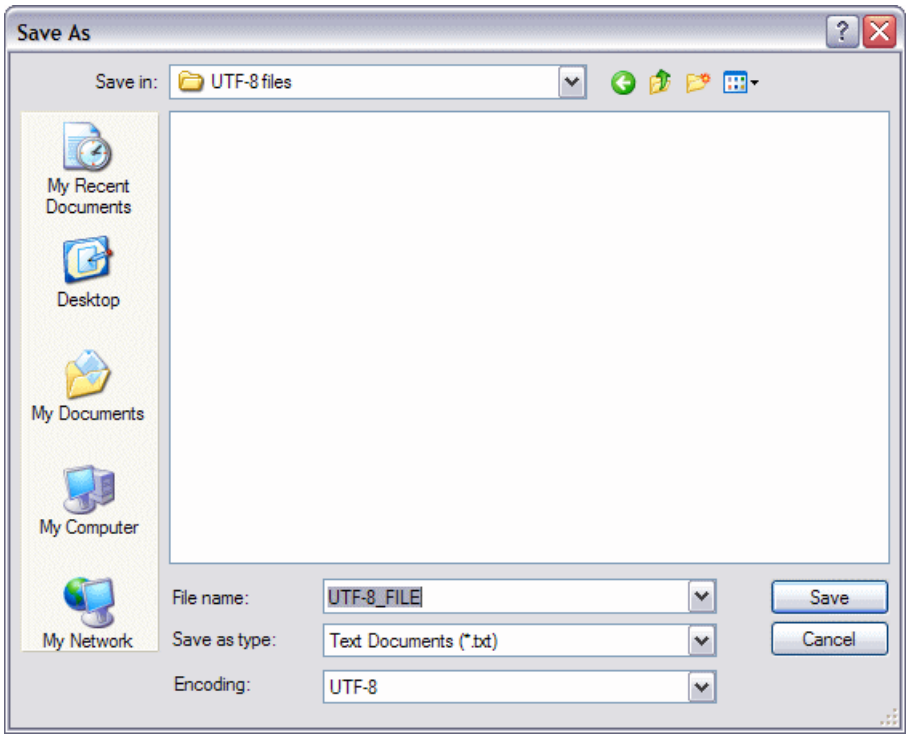


Figure 3-3 UTF-8 file encoding

Example 3-5 illustrates the complete FTP dialog for the file.

Example 3-5 FTP from the Windows client to z/OS FTP Server

```
C:\>ftp 10.1.1.20
Connected to 10.1.1.20.
220-FTPDB1 IBM FTP CS V1R13 at wtsc30.ITSO.IBM.COM, 19:22:28 on 2010-09-30.
220 Connection will close if idle for more than 5 minutes.
User (10.1.1.20:(none)): cs03
331 Send password please.
Password:                                                    1
230 CS03 is logged on.  Working directory is "CS03.".
ftp> ascii                                                   2
200 Representation type is Ascii NonPrint
ftp> quote site encoding=mbcs                                3
200 SITE command was accepted
ftp> quote site mbdataconn=(utf-8,utf-8)                     4
200 SITE command was accepted
ftp> quote site unicodefilesystembom=asis                    5
200 SITE command was accepted
ftp> put c:\UTF-8_FILE.txt 'TCPIP.ITSO.FTPUTF8'              6
200 Port request OK.
125 Storing data set TCPIP.ITSO.FTPUTF8
250 Transfer completed successfully.
ftp: 121 bytes sent in 0.00Seconds 121000.00Kbytes/sec.
ftp>
ftp> quote stat                                              7
211-using Mode Stream, Structure File, type ASCII, byte-size 8
211-ENcoding is set to MBCS
211-MBdataconn codeset names: utf-8,utf-8
211-Server site variable MBREQUIRELASTEOL is set to TRUE
211-Server site variable UNICODEFILESYSTEMBOM is set to ASIS
ftp>
```

In this example, the numbers correspond to the following information:

1. Type in the user password (or password phrase if it is set). It is not shown on the screen.
2. Setting ASCII mode.
3. Setting encoding to mbcs.
4. Setting UTF-8 encoding.
5. Setting the UnicodeFileSystem type.
6. Transferring the UTF-8 file.
7. Displaying the settings to verify proper values.

Dependencies for UTF-8

When a multibyte character set (MBCS) file is created without an EOL <crlf>, setting MBREQUIRELASTEOL on will cause the FTP file transfer to fail, as shown in Example 3-6.

Example 3-6 Attempt to transfer a file without EOL in the last record

```
ftp> put c:\CSFTPUTF 'TCPIP.ITSO.FTPUTF8'
200 Port request OK.
125 Storing data set TCPIP.ITSO.FTPUTF8
451 Transfer aborted due to file error. File is catalogued.                    1
451-File Transfer might be incomplete. Last record received without EOL sequence
ftp: 125 bytes sent in 0.00Seconds 125000.00Kbytes/sec.
```

The number in this example corresponds to the following information:

1. It appears that the file transfer failed, (message 451 implication) but actually the file has been stored in the data set. The problem is that Windows sent the file without an EOL sequence in the final record. To correct this without specifying MBREQUIRELASTEOL, edit the file with an ASCII-based editor and scroll to the end of the data. Press Enter and save the file. Transmitting it again should work. If a file contains an EOL <crlf> before the actual end of the data, FTP transfer will only transmit data up to that EOL and MBREQUIRELASTEOL will return a successful return code.

Note: Because this is an optional feature, there are no migration issues. To take advantage of these functions, you need only use the technique of specifying Unicode operation using MBDATACONN, MBREQUIRELASTEOL, and UNICODEFILESYSTEMBOM. These can be used through the SITE and LOCSITE commands, without changing FTP.DATA.

FTP Kerberos single sign-on support

The FTP server supports single sign-on for Kerberos connections. A Kerberos-based network authentication service enables applications that use Kerberos to use a Kerberos ticket for authentication instead of a user ID and password. This enables you to sign on once to a Kerberos-based security server and not be prompted for a password when accessing the FTP server.

The FTP server statement, SECURE_PASSWORD_KERBEROS with a setting of OPTIONAL permits single sign-on for Kerberos connections.

The FTP client can set the source IP address

The FTP client supports a command line parameter, *-s srcip*, to specify the source IP address that the FTP client should use for client initiated connections to the server. This specification overrides the source IP address that the stack would ordinarily assign to the outbound packets that belong to this client-initiated connection.

TLS and SSL RFC level support

You can configure z/OS FTP to conform to either the draft level or to the RFC 4217 level of securing FTP with TLS. This is specified by coding the TLSRFCLEVEL statement with either a setting of DRAFT or RFC4217.

FTP reporting of incorrect message catalog levels

FTP verifies that the dates within its z/OS UNIX message catalogs, `ftpdmsg.cat` and `ftpdrply.cat`, match the timestamps of the message catalogs used to produce its default message texts. This reduces the chance that FTP will present the wrong message or reply to the user because of a lack of synchronization between message catalogs and FTP load modules.

FTP client support for sequence numbers in the INPUT file

The FTP client, by default, treats sequence numbers present in the INPUT command stream as part of the command. However, these sequence numbers can be removed by the client before processing, by specifying SEQNUMSUPPORT TRUE. This reduces the likelihood that sequence numbers in that input will be interpreted as part of the command to the FTP client. There are specific requirements for this statement when the INPUT consists of multiple, concatenated data sets.

3.2.3 Configuration of basic FTP without security

In this section we set up an FTP server and FTP client with settings we would expect to encounter in a typical installation. We set up the FTP server in one LPAR, the FTP client in a different LPAR, and then use the FTP client to connect to the FTP server.

To use FTP, perform the following tasks.

- Prepare the client environment
- Prepare the server environment

Prepare the client environment

In the LPAR used for the FTP client, you need to create an FTP.DATA for the client. We copied the sample FTP.DATA for the client from SEZAINST(FTCDATA) to our TCPIPB.TCPPARMS. Next, determine if you need to further customize FTCDATA for your installation. *z/OS Communications Server: IP Configuration Reference*, SC31-8776, contains a description of all the FTP.DATA parameters. We used the supplied FTCDATA without any additional changes.

The FTP client uses the search order documented in *z/OS Communications Server: IP Configuration Reference*, SC31-8776, to locate its FTP.DATA data set. We suggest that you use a SYSFTPD DD card in your TSO logon procedure, or use the -f parameter to point to the desired configuration file. This ensures that you are using the correct FTP.DATA data set. Your FTP client is now ready for use.

Prepare the server environment

Perform the remaining tasks on the LPAR used for your FTP server:

- Define SAF definitions for the FTP server
- Create the FTP.DATA for the server
- Create the catalogued procedure for the FTP server
- Create the STDENV file for the FTPD server
- Add FTP to the AUTOLOG and PORT statements
- Update the FTP port entry in /etc/services
- Configure syslogd

Define SAF definitions for the FTP server

At a minimum, the following definitions are required to start and use FTP:

- The FTP catalogued procedure must be defined to the security program and added to the started class facility or started procedures table. The user ID associated with the task must have a UID of 0. If the FACILITY class is active, then the FTP user ID requires READ access to the BPX.DAEMON or BPX.POE profiles, if defined. If the FTP address space is to be nonswappable, then the user ID also requires READ access to FACILITY class resource BPX.STOR.SWAP. We used the same ID used by the TCP/IP stack to start FTPD.
- Any user that will log into FTP must have an OMVS segment defined in the security profile for that user ID.

Important: There might be additional security requirements in your environment if EZB.STACKACCESS, EZB.PORTACCESS, or EZB.NETACCESS controls are in use. *z/OS Communications Server: IP Configuration Guide*, SC31-8775, lists the optional security requirements. Access controls are discussed in *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

Create the FTP.DATA for the server

First, copy the sample FTP.DATA for the server from SEZAINST(FTPSDATA) to your TCPPARMS. Next, determine if you need to further customize FTPSDATA for your installation. The *z/OS Communications Server: IP Configuration Reference*, SC31-8776, contains a description of all the FTP.DATA parameters. We used the sample FTPSDATA member to create our own member called FTPDBxx (where xx is the &sysclone system symbolic), and simply uncommented the BANNER and ADMINEMAILADDR statements. The changes we made to the file for our FTP server installation are shown in Example 3-7. Neither parameter is crucial to the execution of the server, we used them to simply show how to make changes to the file.

Example 3-7 Changes to FTPDBxx which was copied from FTPSDATA

```
BANNER /etc/ftpbanner
ADMINEMAILADDRESS user@host.my.net
```

The contents of our `/etc/banner` file are shown in Example 3-8. Note the `/etc/ftpbanner` file needs to be readable by the user ID of the FTP server proc. The user ID starting FTP is a root user so file permissions 700 (rwx------) provide sufficient access. In our case, this user ID is TCPIP.

Example 3-8 /etc/ftpbanner

```
****************************************
 Welcome to the ITSO FTP server

 This system may be used for
management approved purposes only.

 All transfers are logged.

 Please report problems to %E
****************************************
```

Create the catalogued procedure for the FTP server

Copy the sample FTP procedure from SEZAINST(FTPD) to your PROCLIB and customize the sample. The parameters on the EXEC and the SYSFTPD DD statements need to be updated. Ensure that SYSFTPD refers to the correct FTP.DATA for the server and that the RESOLVER_CONFIG environment variable points to the resolver configuration data set you want to use. We chose to use the _CEE_ENVFILE environment variable to point to a DD statement which defines a data set containing other environment variables for the server. Our procedure is shown in Example 3-9.

Example 3-9 Catalogued procedure for the FTP server

```
//FTPDB  PROC PARMS='',STDENV=FTPENV&SYSCLONE.,FTPDATA=FTPDB&SYSCLONE. 1
//FTPDB  EXEC PGM=FTPD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/&PARMS')
//STDENV  DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV.)   2
//SYSFTPD DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&FTPDATA.)            3
//CEEDUMP  DD SYSOUT=*
```

The following items explain Example 3-9 on page 163:

1. The STDENV symbolic parameter specifies the member name containing environment variables. We made use of the &SYSCLONE system symbolic to determine the name of the member. The FTPDATA symbolic parameter specifies the member name of the FTP.DATA configuration file. We made use of the &SYSCLONE system symbolic to determine the name of the member. Using symbolic parameters on the PROC statement enables more flexibility in starting the procedure.
2. The standard environment variable file must have RECFM =V or VB. The specified member contains environment variable settings.
3. The FTPDATA member contains FTP client configuration statements to override internal defaults.

Create the STDENV file for the FTPD server

Example 3-10 shows the STDENV environment variable file we used for our FTPD server.

Example 3-10 FTP server standard environment variable file

```
BROWSE    TCPIP.SC31.STDENV(FTPENV31) - 01.00        Line 00000000 Col 001 076
******************************** Top of Data ********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'  1
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB              2
_BPX_JOBNAME=FTPDB                            3
TZ=EST                                        4
******************************* Bottom of Data ******************************
```

The following items explain Example 3-10:

1. The resolver needs access to the TCPDATA file for functions such as DNS name resolution.
2. We set stack affinity to our local stack. This is necessary only if multiple stacks exist on the LPAR.
3. The initial job name of the parent process is always FTPDB1 (the proc name suffixed with the number 1) regardless of the job name setting. We set _BPX_JOBNAME so that all subsequent FTP forked instances that are created by a client connection have the *same* job name: FTPDB. Setting the job name in this way enables syslogd *isolation to function* and sends all FTP log messages to a single file, as discussed in Chapter 1, “The syslog daemon” on page 1. If you do not set the _BPX_JOBNAME environment variable, then at each subsequent fork, the job name becomes FTPDB*x,* where *x* is a number between two and nine.
4. We set the TZ environment variable so that messages issued to the console or to SYSLOGD are in the correct time zone.

   We point the server to the FTP.DATA configuration file that we prepared.

Note: You can use the *_CEE_ENVFILE* environment variable in the PARM field of the JCL to point to a file that contains other environment variables. The file can be a UNIX file, a zFS, or a z/OS MVS data set. When it is an MVS data set, the data set must be allocated with RECFM=V.

RECFM=F must *not* be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in z.OS UNIX. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Add FTP to the AUTOLOG and PORT statements

AUTOLOG allows you to automatically start the FTP server when the TCP/IP stack has initialized, and to automatically restart it anytime the FTP server fails. To enable this support you need to add an AUTOLOG statement for the FTP server. AUTOLOG needs to know the name of the procedure that starts FTPD and needs to know the job name of the running FTP server process to monitor it. The FTP server forks on startup, so the actual job name of the running FTP server might be different from the original job name assigned when the FTP server was started. In our environment, we start the FTP server with a job name of FTPDB, and expect the first forked task to be named FTPDB1. FTPDB1 is the name of the FTP server listener. All subsequent FTP server address spaces that represent logged-in users will be named FTPDB because we have set the _BPX_JOBNAME variable to force all FTP jobs to be named FTPDB.

The PORT statement in PROFILE.TCPIP should reserve the FTP control and FTP data ports to the FTP job name. The default ports for FTP are TCP port 21 for the control connection and TCP port 20 for the data connection. The port must be reserved to the job name of the running FTP server. Example 3-11 shows our AUTOLOG and PORT statements in PROFILE.TCPIP.

Example 3-11 Statements in PROFILE.TCPIP for the FTP server

```
AUTOLOG
  FTPD JOBNAME FTPDB1

PORT
  20 TCP  FTPDB NOAUTOLOG
  21 TCP  FTPDB1
```

Update the FTP port entry in /etc/services

The /etc/services maps service names to port names. Unless overridden by a command line parameter, when the FTP server starts it searches `/etc/services` for the service name *ftp* and attempts to bind the FTP control socket to the port associated with the FTP service. Our `/etc/services` is shown in Example 3-12.

Example 3-12 FTP entry in /etc/services

```
ftp  21/tcp
```

Configure syslogd

We recommend that you configure syslogd before starting the FTP server. If you use our recommended syslogd configuration, then your syslogd configuration file already contains one of the lines shown in Example 3-13. This will capture all messages from the FTP server task that has a job name that starts with FTPDB to a separate log file named `ftpd.log`. See Chapter 1, “The syslog daemon” on page 1, for more details about syslogd.

Example 3-13 Line in /etc/syslog.conf for the FTP server

```
*.FTPDB*.*.* /var/log/%Y/%m/%d/ftpd.log              or
*.FTPDB*.*.* /tmp/syslog/%Y/%m/%d/ftpd.log      or
*.FTPDB*.*.* /tmp/syslog/ftpd.log
```

3.2.4 Activation and verification for basic FTP without security

When all the preparation tasks have been completed, you are ready to start the FTP server using your started procedure.

Start the FTP server

To start the FTP server, use this command:

```
S FTPDB
```

There are a number of ways to verify that the FTP server has started and is working correctly. First, we look for message EZY2702I on the system console.

Check startup message for FTP server

Example 3-14 shows the EZY2702I message we received shortly after FTP was started.

Example 3-14 EZY2702I message received on successful startup of FTP

```
EZY2702I Server-FTP: Initialization completed at 19:02:54 on 09/30/10.
```

Verify that FTP server is functional before client login

To verify that the FTP server is functional, we used the z/OS FTP client and tested the FTP client also. The following tasks show the progression of a client connection and the assignments of the job name and user security context to the forked processes. In sequence, we show:

1. A client connection log before user login (before user ID and password submitted)
2. An OMVS display of the FTPDB* threads before user login
3. An MVS display of the active FTPDB* jobs
4. A NETSTAT connection display of the FTPDB* client connections

Check client log for successful connection to server, before client login

Example 3-15 shows the output log of our FTP client from another z/OS LPAR immediately after a connection, but *before* we logged in to the server.

Example 3-15 FTP client log before client login

```
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPA
EZA1554I Connecting to:   10.1.1.20 port: 21.
220-FTPDB1 IBM FTP CS V1R13 at wtsc30.ITSO.IBM.COM, 20:08:44 on 2010-09-30.  1
220 Connection will close if idle for more than 5 minutes.
EZA1459I NAME (10.1.1.20:CS03):                                              2
```

The numbers in this example correspond to the following information:

1. The 220 server message indicates that the server has accepted the connection

2. The client is prompting the user for the user ID to be passed on to the server

Check OMVS status of FTPDB threads before client login*

Example 3-16 shows the OMVS status of the server listener and the client connection *before* the client supplies user ID and password.

Example 3-16 OMVS display of FTPDB threads before client login

```
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
TCPIP    FTPDB1   0057     131322          1 1FI----- 16.52.16    .033  1
  LATCHWAITPID=         0 CMD=FTPD

TCPIP    FTPDB    006C   50462957     131322 1FI----- 17.27.38    .021  2
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 0 0 27 1 80 128 256 35
```

The numbers in this example correspond to the following information:

1. This line shows information for the server's listener:

- The security context (user ID) is TCPIP and always will be because it is the listener's
- The job name is FTPDB1 and always is because it is the first forked process
- The process ID (PID) is 131322
- The time the address space was created is 16.52.16
- The CPU time accumulation in seconds is .033

2. This line shows information for the client connection:

- The security context (user ID) is TCPIP because the client has *not* logged in, yet
- The job name is FTPDB because we used _BPX_JOBNAME to force all forked processes to have the same name
- Notice that this entry has its own PID, but also is tied to the Parent PID (PPID) of the server's listener process (131322)
- The time the address space was created is 17.27.38
- The CPU time accumulation in seconds is .021

Check MVS status of FTPDB jobs before client login*

Example 3-17 shows the output from an MVS display of the FTPDB* jobs *before* client login. Both the FTP server listener task and our forked task for the connected client are shown.

Example 3-17 MVS display of active FTPDB jobs before client login*

```
JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
00009    00029    00001      00035    00025    00001/00075       00034
 FTPDB1   STEP1    TCPIP    OWT  AO  A=0056   PER=NO   SMC=000          1
                                    PGN=N/A  DMN=N/A  AFF=NONE
                                    CT=000.035S  ET=09.47.14            2
                                    WUID=STC03780 USERID=TCPIP           3
                                    WKL=SYSTEM   SCL=SYSOTHER P=1
                                    RGP=N/A      SRVR=NO  QSC=NO
                                    ADDR SPACE ASTE=7E656580
 FTPDB    STEP1    TCPIP    OWT  AO  A=0054   PER=NO   SMC=000          4
                                    PGN=N/A  DMN=N/A  AFF=NONE
                                    CT=000.006S  ET=025.257S            5
                                    WUID=STC05313 USERID=TCPIP           6
                                    WKL=SYSTEM   SCL=SYSOTHER P=1
                                    RGP=N/A      SRVR=NO  QSC=NO
                                    ADDR SPACE ASTE=7E656500
```

The numbers in this example correspond to the following information:

1. Job name FTPDB1 running under user ID TCPIP is the FTP server listener. This is always the case, because it is the first forked process.

2. The Elapsed Time field (ET=) indicates *how long* the server's listener connection has been in place.

3. The USERID field indicates the user ID under which the listener connection is running.

4. Job name FTPDB running under user ID TCPIP is the FTP client connection. This is the forked task that is serving our connected client. It shows the server's user ID until the client logs in with user ID and password, at which time, the security context of the address space will be changed from the server's user ID to the client's user ID

5. The Elapsed Time field (ET=) indicates *how long* the client's connection has been in place.

6. The USERID field indicates the user ID under which the client connection is running. The user ID (security context) will remain the user ID of the FTP server until the client logs in.

Check NETSTAT conn status of FTPDB connections before client login*

Example 3-18 shows the NETSTAT output for connections belonging to FTPDB* *before* client login.

Example 3-18 NETSTAT display of FTPDB connections before client login*

```
D TCPIP,TCPIPB,N,CONN,CLIENT=FTPDB*
EZD0101I NETSTAT CS V1R13 TCPIPB 306
USER ID  CONN     STATE
FTPDB1   00000285 LISTEN                  1
  LOCAL SOCKET:   ::..21                  2
  FOREIGN SOCKET: ::..0
FTPDB1   000003AF ESTBLSH                 3
  LOCAL SOCKET:   ::FFFF:10.1.1.20..21    4
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1137  5
2 OF 2 RECORDS DISPLAYED
```

Job name FTPDB1 shows two connections: 1 The listener, on port 21 (2) in a listen state that represents the listener task. 3 The connection between local port 21 (4) and 10.1.2.10 port 1137 (5) that represents the established control connection for our client. Note that the established connection for the client shows job name FTPDB1 (the server listener ID) and not job name FTPDB, as we would expect. This is because the job that originally accepted the connection was FTPDB1 and only *after* the connection was established was a new job forked to handle the FTP session. So, as far as the stack is concerned, the original job name is what is assigned to the client's control connection. When the data connection is opened for a data transfer, the data connection (on port 20) will be assigned a job name of FTPDB, as expected.

Verify that FTP server is functional after client login

We show the same displays again, *after* user login:

- A client connection log after user login
- An OMVS display of the FTPDB* threads after user login
- An MVS display of the active FTPDB* jobs
- A NETSTAT connection display of the FTPDB* client connections

Check client log after client login

Example 3-19 shows the output log of our FTP client from the other z/OS LPAR(SC30) *after* we successfully logged in to our FTP server (on SC31).

Example 3-19 FTP connect from one z/OS system to another z/OS system: after login

```
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPA
EZA1554I Connecting to:   10.1.1.20 port: 21.
220-FTPDB1 IBM FTP CS V1R13 at wtsc30.ITSO.IBM.COM, 21:02:04 on 2010-09-30.  1
220 Connection will close if idle for more than 5 minutes.
EZA1459I NAME (10.1.1.20:CS03):
cs03
EZA1701I >>> USER cs03
331 Send password please.
EZA1789I PASSWORD:
EZA1701I >>> PASS

230 CS03 is logged on.  Working directory is "CS03.".                      2
EZA1460I Command:
```

The numbers in this example correspond to the following information:

1. The 220 server message indicates that the server on SC31 has accepted the connection.

2. The 230 server message indicates that the server on SC31 has accepted the user ID and password from the client, and has established the default high level qualifier for data set access to that of the user ID.

Check OMVS status of FTPDB threads after client login*

Example 3-20 shows the OMVS status of the server listener and the client connection *after* the client supplies user ID and password.

Example 3-20 OMVS display of FTPDB threads after client login

```
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
TCPIP    FTPDB1   0057     131322          1 1FI----- 16.52.16    .033  1
  LATCHWAITPID=         0 CMD=FTPD
CS03     FTPDB    006C   50462957     131322 1FI----- 17.27.38    .041  2
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2136115088
```

The numbers in this example correspond to the following information:

1. This line shows information for the server's listener: (it has not changed)

- The security context (user ID) is TCPIP and always will be because it is the listener's
- The job name is FTPDB1 and always is because it is the first forked process
- The process ID (PID) is 131322
- The time the address space was created is 16.52.16
- The CPU time accumulation in seconds is .033

2. This line shows information for the client connection: (some information has changed)

- The security context has been changed to CS10, the client has logged in
- The job name is FTPDB because we used _BPX_JOBNAME to force all forked processes to have the same name
- Notice that this entry has its own PID, but also is tied to the Parent PID (PPID) of the server's listener process (131322)
- The time the address space was created is 17.57.38
- The CPU time accumulation in seconds has changed from .021 to .041, which is attributed to the login process.

Check MVS status of FTPDB jobs after client login*

Example 3-21 shows output from the D J,FTPDB* command *after* we logged into the FTP server. Both the FTP server listener task and our forked task for the connected client are shown.

Example 3-21 D J,FTPDB after login to FTP server*

```
JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
00009    00029    00001      00035    00025    00001/00075       00034
 FTPDB1   STEP1    TCPIP    OWT  AO  A=0057   PER=NO   SMC=000          1
                                     PGN=N/A  DMN=N/A  AFF=NONE
                                     CT=000.026S  ET=09.49.41           2
                                     WUID=STC03781 USERID=TCPIP         3
                                     WKL=SYSTEM   SCL=SYSOTHER P=1
                                     RGP=N/A      SRVR=NO  QSC=NO
                                     ADDR SPACE ASTE=7E6565C0
 FTPDB    STEP1    CS03     OWT  AO  A=006C   PER=NO   SMC=000          4
                                     PGN=N/A  DMN=N/A  AFF=NONE
```

```
CT=000.007S  ET=044.230S          5
WUID=STC05219 USERID=CS03         6
WKL=SYSTEM   SCL=SYSOTHER P=1
RGP=N/A      SRVR=NO  QSC=NO
ADDR SPACE ASTE=7E656B00
```

The numbers in this example correspond to the following information:

1. Job name FTPDB1 running under user ID TCPIP is the FTP server listener.
2. The Elapsed Time field (ET=) indicates *how long* the server's listener connection has been in place.
3. The USERID field indicates the user ID under which the listener connection is running.
4. Job name FTPDB running under user ID CS03 is the FTP client connection. This is the forked task that is serving our connected client.
5. The Elapsed Time field (ET=) indicates *how long* the client's connection has been in place.
6. The USERID field indicates the user ID under which the client connection is running.

Check NETSTAT conn status of FTPDB connections after client login*

Example 3-22 shows the output from a **`netstat conn`** command after client login.

Example 3-22 Output of netstat conn, after user login

```
D TCPIP,TCPIPB,N,CONN,CLIENT=FTPDB*
EZD0101I NETSTAT CS V1R13 TCPIPB 306
USER ID  CONN     STATE
FTPDB1   00000285 LISTEN                 1
  LOCAL SOCKET:   ::..21                 2
  FOREIGN SOCKET: ::..0
FTPDB1   000003AF ESTBLSH                3
  LOCAL SOCKET:   ::FFFF:10.1.1.20..21   4
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1137 5
2 OF 2 RECORDS DISPLAYED
```

Job name FTPDB1 shows two connections: 1 The listener, on port 21 (2) in a listen state that represents the listener task. 3 The connection between local port 21 (4) and 10.1.2.10 port 1137 (5) that represents the established control connection for our client. Note that the established connection for the client shows job name FTPDB1 (the server listener ID) and not job name FTPDB, as we would expect. This is because the job that originally accepted the connection was FTPDB1 and only *after* the connection was established was a new job forked to handle the FTP session. So, as far as the stack is concerned, the original job name is what is assigned to the client's control connection. When the data connection is opened for a data transfer, the data connection (on port 20) will be assigned a job name of FTPDB, as expected. See Example 3-23 on page 171.

Check NETSTAT conn status of FTPDB connections during file transfer*

Example 3-23 shows the output from a **netstat conn** command during a client file transfer. When the data connection for the transfer is opened, it will be on port 20, as seen here.

Example 3-23 Output of netstat conn, during a client file transfer

```
D TCPIP,TCPIPB,N,CONN,CLIENT=FTPDB*
. . .
FTPDB    00000397 ESTBLSH                1
  LOCAL SOCKET:   ::FFFF:10.1.1.20..20   2
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1141 3
FTPDB1   00000285 LISTEN                 4
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDB1   000003AF ESTBLSH                5
  LOCAL SOCKET:   ::FFFF:10.1.1.20..21
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1137
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

The numbers in this example correspond to the following information:

1. The data connection on port 20 is being used for the file transfer and has a job name of FTPDB which was assigned by the *_BPX_JOBNAME* setting. All forked processes will have this common job name.
2. The local socket is the IP address of this system that the client used to access this system. Port 20 is used for the data connection.
3. The foreign socket is the IP address of the client that is using ephemeral port 1141 for its data connection.
4. The same original server listener connection on port 21, listening on INADDR_ANY
5. The same original client control connection on port 21 as before

Check server parameter settings with the STAT command

The STAT command is issued by the client to request a listing of the server options. The server responds with reply code 211 messages. We show an abbreviated list of the server options in Example 3-24.

Example 3-24 STAT command shows server option settings

```
>>> STAT
211-Server FTP talking to host ::ffff:10.1.2.10, port 1137          1
211-User: CS03  Working directory: CS03.                            2
211-The control connection has transferred 288 bytes                3
211-There is no current data connection.
. . .
211-RDWs from variable format data sets are discarded.              4
. . .
211-ENcoding is set to SBCS                                         5
211-Outbound SBCS ASCII data uses CRLF line terminator
211-Outbound MBCS ASCII data uses CRLF line terminator
211-Server site variable MBREQUIRELASTEOL is set to TRUE
211-Server site variable UNICODEFILESYSTEMBOM is set to ASIS
. . .
211-TLS security is supported at the DRAFT level                    6
. . .
211-Server site variable LISTSUBDIR is set to TRUE                  7
211 *** end of status ***
```

The numbers in this example correspond to the following information:

1. The server confirms the client's IP address and port being used
2. The user ID and current working directory are indicated
3. Information the control connection and the data connection is listed
4. Information on how RDWs are treated for RECFM=VB records
5. Single-byte and multi-byte encoding settings are available for display
6. TLS security level support is shown (TLSRFCLEVEL is DRAFT or RFC4217)
7. LISTSUBDIR controls how file names within subdirectories are displayed

Check client parameter settings with the LOCSTAT command

The LOCSTAT command is issued by the client to request a listing of the local client options. The client program responds with a list of option settings. We show an abbreviated list of the client options in Example 3-25.

Example 3-25 LOCSTAT command shows client option settings

```
>>> LOCSTAT
 Trace: FALSE, Send Port: TRUE
 Send Site with Put command: TRUE
 Connected to:10.1.1.20, Port: FTP control (21), logged in                1
 Local Port: 1137
 Data type:a, Transfer mode:s, Structure:f
. . .
 Data connections for the client are not firewall friendly.
 local site variable EPSV4 is set to FALSE
 local site variable SECUREIMPLICITZOS is set to TRUE
 local site variable TLSRFCLEVEL is set to DRAFT                          2
 local site variable LISTSUBdir is set to TRUE
 local site variable PROGRESS is set to 10
 local site variable SEQNUMSUPPORT is set to FALSE                        3
 Authentication mechanism: None
. . .
 Using FTP configuration defaults.                                        4
```

The numbers in this example correspond to the following information:

1. The current connection information includes the IP address and port number of the server (10.1.1.20.21), and the local ephemeral port of the client (1031).
2. The TLSRFCLEVEL setting at the client can be either DRAFT or RFC4217
3. SEQNUMSUPPORT indicates whether the client supports the presence of sequence numbers in the INPUT command stream. If supported (TRUE), the client program removes them before processing the data. If not supported (FALSE), the sequence numbers are assumed to be part of the input commands.
4. This line lets you know from what source the FTP client initially gets its option settings. If an FTP.DATA file was specified to the client, its data set name is listed; otherwise internal hardcoded defaults are used for all the client options. The LOCSITE command can override the initial settings during the session.

3.3 Multiple FTP servers in a sysplex

This section provides an overview of executing multiple FTP servers in a sysplex (one server per LPAR, one stack per LPAR) and using sysplex distribution to load balance between them. Based upon installation policies, the sysplex distributor directs connections to the best available FTP server. We discuss the following topics:

- Description of multiple FTP servers in a sysplex
- Configuration for multiple FTP servers in the sysplex
- Activation and verification of FTP servers within sysplex

3.3.1 Description of multiple FTP servers in a sysplex

For this situation, we use two system images, each with only one TCP/IP stack and only one FTP server associated with it. The TCP/IP stack started-task names are TCPIPA and TCPIPB. The FTPD server started-task names are FTPDA and FTPDB. We use system symbolics in the started task JCL to provide uniqueness where necessary.

On system SC30 we have:

- TCPIPA
- FTPDB

On system SC31 we have:

- TCPIPB
- FTPDB

We designed the two stacks to back up each other. We also designed the two FTP servers to back up each other. Even though it is not a requirement for a backup stack to distribute connections identically to the method used in the primary stack, we designed our two stacks to do so. If the primary stack fails, or otherwise relinquishes its distributor responsibilities, our backup stack will continue to distribute connections to the FTPD servers in the same fashion as the primary.

Dependencies of multiple servers within a sysplex

To use FTP in a sysplex distributor environment, you need to configure FTP on each LPAR in the sysplex using the directions discussed in 3.2, “Basic FTP without security” on page 148.

Stack dependencies for multiple FTPD servers in sysplex

Because sysplex distribution (SD) is being used in this scenario, all the functionality that a TCP/IP stack needs to support SD is required. See *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998, for information about setting up a TCP/IP stack to support sysplex distribution. These requirements are as follows:

- The system hardware and software required to support the coupling facility and XCF.
- XCFINIT=YES in VTAM, DYNAMICXCF in TCP/IP.
- An IP subnet and host IP address assigned to the XCF interfaces.
- If HiperSockets are implemented, HiperSockets used by XCF must be consistent in their definitions
- Most of the parameters within GLOBALCONFIG, IPCONFIG, TCPCONFIG, and UDPCONFIG should be set the same on all participating stacks.

The multiple stacks must have Distributed Dynamic VIPA definitions added to support distribution to the multiple FTPD servers. The VIPADYNAMIC block must be coded in the backup stack in such a way that it will distribute connections in a similar manner as the primary, but not necessarily in an identical manner. Our scenario calls for an identical process.

The introduction of sysplex distribution adds the requirement for a new IP subnet for Distributed DVIPA as though Dynamic VIPA has not already been implemented.

FTP server dependencies for multiple servers in sysplex

Because the two FTP servers back each other up, they should be identically configured so they can treat all client connections the same. If they differ in any way clients could experience differences between their FTP sessions.

Advantages of multiple servers within a sysplex

Sysplex distribution provides multiple redundant resources that provide high availability. In our scenario all of the following are redundant resources participating in the high availability that sysplex distribution provides:

- System images
- Sysplex links
- TCP/IP stacks
- Stack interfaces
- Server applications (FTP servers in this case)

Sysplex distribution working with Workload Manager provides load balancing between the stacks and between the FTPD servers.

For more on the advantages of high availability and work load balancing, see those discussions in the following resources:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Considerations for running multiple servers within a sysplex

Implementing multiple redundant TCP/IP stacks and multiple redundant FTP servers adds to the clerical effort of systems personnel in maintaining equivalent configurations across all participating systems. The effort in keeping the configurations synchronized is sometimes underestimated or overlooked.

Planning and design is more complex, involving multiple departments. The necessity for cooperation between departments is sometimes underestimated or even unplanned. Mainframe systems and networking personnel must be aware of the physical network requirements.

The demand on IP subnets and IP addressing is increased by introducing sysplex distribution, Dynamic VIPAs, and Dynamic XCF.

Operations must be made aware of changes that sysplex distribution, multiple stacks, and multiple server applications introduce to the environment.

Automations and scheduling changes might be required. These issues are sometimes overlooked.

3.3.2 Configuration for multiple FTP servers in the sysplex

Multiple FTP servers can accept connections that are distributed by sysplex distribution. In our environment we set up a sysplex using two stacks, with the job names TCPIPA and TCPIPB that are in two LPARS, SC30 and SC31. The same sysplex environment is discussed in more detail in 2.3, “Multiple TN3270E servers in a multiple image environment” on page 77.

To load balance FTP in a sysplex, you need to first perform all the implementation tasks for both client and server, as described in 3.2.3, “Configuration of basic FTP without security” on page 162. Alternatively, if security is needed, perform all the secure FTP tasks in *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999. Then, perform the following additional tasks:

- Customize a second TCP/IP stack started task procedure.
- Figure .
- Customize a second FTP server started task procedure.
- Customize a second FTP server configuration data set.
- Customize an OMPROUTE started task to support the second stack.
- Customize an OMPROUTE configuration for the second OMPROUTE.

Customize a second TCP/IP stack started task procedure

This second started task is for running our sysplex distribution backup stack. It can be modeled after our first (primary) started task. We used the same procedure name and same procedure library for both stacks, and used system symbolics to provide unique names for the stacks’ configuration profile data sets. Our first TCP/IP task is TCPIPB in LPAR SC31 and our second TCP/IP task is also named TCPIPA and is running in LPAR SC30. The TCP/IP started procedures are identical and use system symbolics to identify different TCPIP.DATA and PROFILE.TCPIP data sets.

Customize both TCP/IP stack profile data sets for the sysplex

The stack in LPAR SC31 is our sysplex distributor stack, and the stack in LPAR SC30 is the backup stack. Both stacks are FTP targets. For a complete discussion and examples of setting up a stack, see the following sources:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Add statements that enable sysplex functions, create dynamic XCF interfaces, and create dynamic VIPAs. The statements necessary to enable sysplex distribution on TCPIPB on LPAR SC31 are shown in Example 3-26.

Example 3-26 Sysplex enablement for TCPIPB on SC31

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60                    1
;
IPCONFIG
   SYSPLEXROUTING                                                     2
   DYNAMICXCF 10.1.7.21 255.255.255.0 8                               3
.
VIPADYNAMIC

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
```

```
 VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.25   ;FTP            4
 VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM                5
                       10.1.8.25    PORT 20 21
                DESTIP 10.1.7.11
                       10.1.7.21

;------------------------------------------------------------------
;  Set up Sysplex Distribution for FTP using ROUNDROBIN         -
;------------------------------------------------------------------
 VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21  ;FTP General
 VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                       10.1.8.21   PORT 20 21
                DESTIP 10.1.7.11
                       10.1.7.21

;------------------------------------------------------------------
;  Set up Sysplex Distribution for FTP using BASEWLM            -
;------------------------------------------------------------------
 VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22  ;FTP Admin
 VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                       10.1.8.22   PORT 20 21
                DESTIP 10.1.7.11
                       10.1.7.21

;------------------------------------------------------------------
;  Set up Sysplex Distribution for FTP using SERVERWLM          -
;------------------------------------------------------------------
 VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23  ;FTP Payroll
 VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                       10.1.8.23   PORT 20 21
                DESTIP 10.1.7.11
                       10.1.7.21
;------------------------------------------------------------------
;      Distribute to 10.1.7.21 via normal XCF   (no viparoute)  -
;      Distribute to 10.1.7.11 via IP routing   (   viparoute)  -
;------------------------------------------------------------------
VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa      6
;VIPAROUTE     DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa

ENDVIPADYNAMIC
```

In this example, the numbers correspond to the following information:

1. SYSPLEXMONITOR DelayJoin keeps the stack from joining the sysplex until after OMPROUTE has completely initialized so that routing can be in full effect.
2. SYSPLEXROUTING enables sysplex distribution.
3. DYNAMICXCF assigns this stack to the XCF subnet and instructs the stack to participate in the sysplex
4. VIPADEFINE establishes the FTP VIPA address which clients use to access FTP
5. VIPADISTRIBUTE sets how to distribute connections to the FTP servers
6. VIPAROUTE is used to offload XCF by redirecting distributed traffic away from the XCF interfaces and toward the other available IP interfaces which are more efficient and usually faster.

The statements necessary to enable sysplex distribution in TCPIPA on LPAR SC30 are shown in Example 3-27.

Example 3-27 Sysplex enablement for TCPIPA on SC30

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.1.7.11 255.255.255.0 8
;
VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.25   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25    PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.21  ;FTP General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.1.8.21   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.22  ;FTP Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.1.8.22   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.23  ;FTP Payroll
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;      Distribute to 10.1.7.21 via IP routing   (   viparoute)  -
  ;      Distribute to 10.1.7.11 via normal XCF   (no viparoute)  -
  ;-------------------------------------------------------------------
  ;VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa
  VIPAROUTE      DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa

ENDVIPADYNAMIC
```

Customize a second FTP server started task procedure

This second started task is for our second FTP server. The procedure can be modeled after our first server started task. We used the same procedure name and same procedure library for both servers and used system symbolics to provide unique names for the FTP configuration profile data sets. The catalogued procedure we used to start both FTP servers is shown in Example 3-9 on page 163.

Customize a second FTP server configuration data set

The FTP.DATA for both servers should be identical. The FTP.DATA file we used for both FTP servers is discussed in "Create the FTP.DATA for the server" on page 163.

Customize an OMPROUTE started task to support the second stack

This second started task is for our second OMPROUTE. It can be modeled after our first started task. We used the same procedure name and same procedure library for both servers and used system symbolics to provide unique names for the OMPROUTE configuration data sets. The omproute started task we used is exactly the same as the started task discussed in "Customize an OMPROUTE started task to support the second stack" on page 84.

Customize an OMPROUTE configuration for the second OMPROUTE

This second configuration data set can be modeled after the first. Obviously, there are statements that must be different between the two configurations to give them their uniqueness: interface IP addresses and router IDs are examples. For details about configuring OMPROUTE, see the following sources:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

The omproute configuration we used is exactly the same as the configuration discussed in "Customize an OMPROUTE configuration for the second OMPROUTE" on page 84.

3.3.3 Activation and verification of FTP servers within sysplex

Issue the MVS START command on both systems:

```
S TCPIPB
```

If FTP is autologged in both stacks, then FTP will automatically start and this step can be skipped. Otherwise, issue the MVS START command for the FTPDB job on both systems:

```
S FTPDB
```

All of the verification tasks previously discussed in 3.2, "Basic FTP without security" on page 148 can be used to verify the FTPD server is running correctly in the sysplex. Additionally, a number of NETSTAT displays can be used to show the status of Dynamic and Distributed VIPA connections. The format of the system NETSTAT command is:

```
D TCPIP,TCPIPB,N,command,option
```

For complete details about the NETSTAT command, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

We discuss the following additional tasks to verify sysplex distribution to FTP servers in this section:

- Use NETSTAT VCRT to show dynamic VIPA connection routing table.
- Use NETSTAT VDPT to show dynamic VIPA destination port table.
- Use NETSTAT VIPADCFG to show current dynamic VIPA configuration.
- Use NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE.

Use NETSTAT VCRT to show dynamic VIPA connection routing table

VCRT displays the dynamic VIPA connection routing table information. For each table entry that represents an established dynamic VIPA connection or an affinity created by the passive-mode FTP, the DETAIL suboption additionally displays the policy rule, action information, and routing information. For each entry that represents an affinity created by the TIMEDAFFINITY parameter on the VIPADISTRIBUTE profile statement, it displays the preceding information plus the affinity-related information.

Example 3-28 shows the connections at the time the VCRT command was issued. Notice that the distributing stack knows about all of the connections because it is managing them.

Example 3-28 NETSTAT VCRT on system SC31, the distributing stack

```
RO SC31,D TCPIP,TCPIPB,N,VCRT
   RESPONSE=SC31
    EZD0101I NETSTAT CS V1R13 TCPIPB 466
    DYNAMIC VIPA CONNECTION ROUTING TABLE:
    DEST:      10.1.8.21..21
      SOURCE:  10.1.100.222..1904
      DESTXCF: 10.1.7.11
    DEST:      10.1.8.21..21
      SOURCE:  10.1.100.223..1972
      DESTXCF: 10.1.7.21
    2 OF 2 RECORDS DISPLAYED
    END OF THE REPORT
```

Notice that the non-distributing stack shows only the connections that have been distributed to it, as shown in Example 3-29.

Example 3-29 NETSTAT VCRT on system SC30, the non-distributing stack

```
RO SC30,D TCPIP,TCPIPA,N,VCRT
   RESPONSE=SC30
    EZD0101I NETSTAT CS V1R13 TCPIPA 030
    DYNAMIC VIPA CONNECTION ROUTING TABLE:
    DEST:      10.1.8.21..21
      SOURCE:  10.1.100.222..1904
      DESTXCF: 10.1.7.11
    1 OF 1 RECORDS DISPLAYED
    END OF THE REPORT
```

Use NETSTAT VDPT to show dynamic VIPA destination port table

VDPT displays the dynamic VIPA destination port table information. If the DETAIL suboption is specified, the output contains policy action information, target responsiveness values, and a WQ value (on a separate line).

Example 3-30 shows the port table entries at the time of issuing the VDPT command. SC31 is currently the distributor, so it shows the ports being distributed and whether there is a ready listener on the port.

Note: The TOTALCONN field indicates the total number of connections there have been since the distribution started for the port. It does *not* represent the current number of connections.

Example 3-30 NETSTAT VDPT on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VDPT
   RESPONSE=SC31
     EZD0101I NETSTAT CS V1R13 TCPIPB 472
     DYNAMIC VIPA DESTINATION PORT TABLE FOR TCP/IP STACKS:
     DEST:        10.1.8.21..20
       DESTXCF:   10.1.7.11
       TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
       DISTMETHOD: ROUNDROBIN
       FLG:
     DEST:        10.1.8.21..20
       DESTXCF:   10.1.7.21
       TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
       DISTMETHOD: ROUNDROBIN
       FLG:
     DEST:        10.1.8.21..21
       DESTXCF:   10.1.7.11
       TOTALCONN: 0000000001  RDY: 001  WLM: 01  TSR: 100
       DISTMETHOD: ROUNDROBIN
       FLG:
     DEST:        10.1.8.21..21
       DESTXCF:   10.1.7.21
       TOTALCONN: 0000000004  RDY: 001  WLM: 01  TSR: 100
       DISTMETHOD: ROUNDROBIN
       FLG:
     DEST:        10.1.8.22..20
       DESTXCF:   10.1.7.11
       TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
       DISTMETHOD: BASEWLM
       FLG:
     DEST:        10.1.8.22..20
       DESTXCF:   10.1.7.21
       TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
       DISTMETHOD: BASEWLM
       FLG:
     DEST:        10.1.8.22..21
       DESTXCF:   10.1.7.11
       TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
       DISTMETHOD: BASEWLM
       FLG:
     DEST:        10.1.8.22..21
       DESTXCF:   10.1.7.21
       TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
       DISTMETHOD: BASEWLM
       FLG:
     DEST:        10.1.8.23..20
```

```
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
    DISTMETHOD: SERVERWLM
    FLG:
  DEST:        10.1.8.23..20
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
    DISTMETHOD: SERVERWLM
    FLG:
  DEST:        10.1.8.23..21
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
    DISTMETHOD: SERVERWLM
    FLG:
  DEST:        10.1.8.23..21
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
    DISTMETHOD: SERVERWLM
    FLG:
  DEST:        10.1.8.25..20
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
    DISTMETHOD: BASEWLM
    FLG:
  DEST:        10.1.8.25..20
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
    DISTMETHOD: BASEWLM
    FLG:
  DEST:        10.1.8.25..21
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
    DISTMETHOD: BASEWLM
    FLG:
  DEST:        10.1.8.25..21
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
    DISTMETHOD: BASEWLM
    FLG:
  16 OF 16 RECORDS DISPLAYED
  END OF THE REPORT
```

SC30 is not a distributor at the moment; therefore, it shows no information, as shown in Example 3-31.

Example 3-31 NETSTAT VDPT on system SC30

```
RO SC30,D TCPIP,TCPIPA,N,VDPT
   RESPONSE=SC30
    EZD0101I NETSTAT CS V1R13 TCPIPA 110
    0 OF 0 RECORDS DISPLAYED
    END OF THE REPORT
```

Use NETSTAT VIPADCFG to show current dynamic VIPA configuration

VIPADCFG displays the current dynamic VIPA configuration information from the perspective of the stack on which the command is entered. The primary distributor shows the VIPA DEFINE, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 3-32.

Example 3-32 NETSTAT VIPADCFG on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADCFG
RESPONSE=SC31
 EZD0101I NETSTAT CS V1R13 TCPIPB 491
 DYNAMIC VIPA INFORMATION:
   VIPA DEFINE:
     IPADDR/PREFIXLEN: 10.1.8.21/24
       MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG:
     IPADDR/PREFIXLEN: 10.1.8.22/24
       MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG:
     IPADDR/PREFIXLEN: 10.1.8.23/24
       MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG:
     IPADDR/PREFIXLEN: 10.1.8.25/24
       MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG:
   VIPA DISTRIBUTE:
     DEST:      10.1.8.21..20
       DESTXCF: 10.1.7.11
       DISTMETHOD: ROUNDROBIN
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.21..20
       DESTXCF: 10.1.7.21
       DISTMETHOD: ROUNDROBIN
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.21..21
       DESTXCF: 10.1.7.11
       DISTMETHOD: ROUNDROBIN
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.21..21
       DESTXCF: 10.1.7.21
       DISTMETHOD: ROUNDROBIN
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.22..20
       DESTXCF: 10.1.7.11
       DISTMETHOD: BASEWLM
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.22..20
       DESTXCF: 10.1.7.21
       DISTMETHOD: BASEWLM
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.22..21
       DESTXCF: 10.1.7.11
       DISTMETHOD: BASEWLM
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.22..21
       DESTXCF: 10.1.7.21
       DISTMETHOD: BASEWLM
       SYSPT:   NO   TIMAFF: NO    FLG:
     DEST:      10.1.8.23..20
       DESTXCF: 10.1.7.11
       DISTMETHOD: SERVERWLM
```

```
      SYSPT:   NO   TIMAFF: NO    FLG:
    DEST:      10.1.8.23..20
      DESTXCF: 10.1.7.21
      DISTMETHOD: SERVERWLM
      SYSPT:   NO   TIMAFF: NO    FLG:
    DEST:      10.1.8.23..21
      DESTXCF: 10.1.7.11
      DISTMETHOD: SERVERWLM
      SYSPT:   NO   TIMAFF: NO    FLG:
    DEST:      10.1.8.23..21
      DESTXCF: 10.1.7.21
      DISTMETHOD: SERVERWLM
      SYSPT:   NO   TIMAFF: NO    FLG:
    DEST:      10.1.8.25..20
      DESTXCF: 10.1.7.11
      DISTMETHOD: BASEWLM
      SYSPT:   YES  TIMAFF: NO    FLG:
    DEST:      10.1.8.25..20
      DESTXCF: 10.1.7.21
      DISTMETHOD: BASEWLM
      SYSPT:   YES  TIMAFF: NO    FLG:
    DEST:      10.1.8.25..21
      DESTXCF: 10.1.7.11
      DISTMETHOD: BASEWLM
      SYSPT:   YES  TIMAFF: NO    FLG:
    DEST:      10.1.8.25..21
      DESTXCF: 10.1.7.21
      DISTMETHOD: BASEWLM
      SYSPT:   YES  TIMAFF: NO    FLG:
  VIPA ROUTE:
    DESTXCF:     10.1.7.11
      TARGETIP:  10.1.1.10
END OF THE REPORT
```

The backup stack shows the VIPA BACKUP, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 3-33.

Example 3-33 NETSTAT VIPADCFG on system SC30

```
RO SC30,D TCPIP,TCPIPA,N,VIPADCFG
  RESPONSE=SC30
  EZD0101I NETSTAT CS V1R13 TCPIPA 187
   DYNAMIC VIPA INFORMATION:
     VIPA BACKUP:
       IPADDR/PREFIXLEN: 10.1.8.21
         RANK: 200  MOVEABLE:           SRVMGR:     FLG:
       IPADDR/PREFIXLEN: 10.1.8.22
         RANK: 200  MOVEABLE:           SRVMGR:     FLG:
       IPADDR/PREFIXLEN: 10.1.8.23
         RANK: 200  MOVEABLE:           SRVMGR:     FLG:
       IPADDR/PREFIXLEN: 10.1.8.25
         RANK: 200  MOVEABLE:           SRVMGR:     FLG:
     VIPA DISTRIBUTE:
       DEST:      10.1.8.21..20
         DESTXCF: 10.1.7.11
         DISTMETHOD: ROUNDROBIN
```

```
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.21..20
  DESTXCF: 10.1.7.21
  DISTMETHOD: ROUNDROBIN
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.21..21
  DESTXCF: 10.1.7.11
  DISTMETHOD: ROUNDROBIN
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.21..21
  DESTXCF: 10.1.7.21
  DISTMETHOD: ROUNDROBIN
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.22..20
  DESTXCF: 10.1.7.11
  DISTMETHOD: BASEWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.22..20
  DESTXCF: 10.1.7.21
  DISTMETHOD: BASEWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.22..21
  DESTXCF: 10.1.7.11
  DISTMETHOD: BASEWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.22..21
  DESTXCF: 10.1.7.21
  DISTMETHOD: BASEWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.23..20
  DESTXCF: 10.1.7.11
  DISTMETHOD: SERVERWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.23..20
  DESTXCF: 10.1.7.21
  DISTMETHOD: SERVERWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.23..21
  DESTXCF: 10.1.7.11
  DISTMETHOD: SERVERWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.23..21
  DESTXCF: 10.1.7.21
  DISTMETHOD: SERVERWLM
  SYSPT:   NO   TIMAFF: NO    FLG:
DEST:      10.1.8.25..20
  DESTXCF: 10.1.7.11
  DISTMETHOD: BASEWLM
  SYSPT:   YES  TIMAFF: NO    FLG:
DEST:      10.1.8.25..20
  DESTXCF: 10.1.7.21
  DISTMETHOD: BASEWLM
  SYSPT:   YES  TIMAFF: NO    FLG:
DEST:      10.1.8.25..21
  DESTXCF: 10.1.7.11
```

```
        DISTMETHOD: BASEWLM
        SYSPT:   YES  TIMAFF: NO    FLG:
      DEST:      10.1.8.25..21
        DESTXCF: 10.1.7.21
        DISTMETHOD: BASEWLM
        SYSPT:   YES  TIMAFF: NO    FLG:
    VIPA ROUTE:
      DESTXCF:     10.1.7.21
        TARGETIP:  10.1.1.20
  END OF THE REPORT
```

Use NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE

VIPADYN displays the current dynamic VIPA and VIPAROUTE information from the perspective of the stack on which the command is entered. Two suboptions are available to filter the output:

- DVIPA: Displays the current dynamic VIPA information only
- VIPAROUTE: Displays the current VIPAROUTE information only

Example 3-34 shows SC31 information.

Example 3-34 NETSTAT VIPADYN on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN
   RESPONSE=SC31
    EZD0101I NETSTAT CS V1R13 TCPIPB 496
    DYNAMIC VIPA:
      IPADDR/PREFIXLEN: 10.1.8.21/24
        STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
        ACTTIME: 10/01/2010 18:53:52
      IPADDR/PREFIXLEN: 10.1.8.22/24
        STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
        ACTTIME: 10/01/2010 18:53:52
      IPADDR/PREFIXLEN: 10.1.8.23/24
        STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
        ACTTIME: 10/01/2010 18:53:52
      IPADDR/PREFIXLEN: 10.1.8.25/24
        STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
        ACTTIME: 10/01/2010 18:53:52
    VIPA ROUTE:
      DESTXCF: 10.1.7.11
        TARGETIP: 10.1.1.10
        RTSTATUS: ACTIVE
    5 OF 5 RECORDS DISPLAYED
    END OF THE REPORT
```

Example 3-35 shows the same information for SC31: with filters for dvipa only.

Example 3-35 NETSTAT VIPADYN,DVIPA on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
   RESPONSE=SC31
    EZD0101I NETSTAT CS V1R13 TCPIPB 509
    DYNAMIC VIPA:
      IPADDR/PREFIXLEN: 10.1.8.21/24
        STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
```

```
      ACTTIME: 10/01/2010 18:53:52
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 10/01/2010 18:53:52
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 10/01/2010 18:53:52
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 10/01/2010 18:53:52
  4 OF 4 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 3-36 shows SC31 Netstat output for VIPADYN,VIPAROUTE, with filters for viparoute only.

Example 3-36 NETSTAT VIPADYN,VIPAROUTE on system SC30

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
   RESPONSE=SC31
    EZD0101I NETSTAT CS V1R13 TCPIPB 520
    VIPA ROUTE:
      DESTXCF: 10.1.7.11
        TARGETIP: 10.1.1.10
        RTSTATUS: ACTIVE
    1 OF 1 RECORDS DISPLAYED
    END OF THE REPORT
```

Example 3-37 shows SC30Netstat output for VIPADYN ftp and FTP dvipa addresses.

Example 3-37 NETSTAT VIPADYN on system SC31

```
RO SC30,D TCPIP,TCPIPA,N,VIPADYN
   RESPONSE=SC30
    EZD0101I NETSTAT CS V1R13 TCPIPA 320
    DYNAMIC VIPA:
      IPADDR/PREFIXLEN: 10.1.8.21/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
      IPADDR/PREFIXLEN: 10.1.8.22/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
      IPADDR/PREFIXLEN: 10.1.8.23/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
      IPADDR/PREFIXLEN: 10.1.8.25/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
    VIPA ROUTE:
      DESTXCF: 10.1.7.21
        TARGETIP: 10.1.1.20
        RTSTATUS: ACTIVE
    5 OF 5 RECORDS DISPLAYED
    END OF THE REPORT
```

Example 3-38 shows SC30 Netstat output for VIPADYN,DVIPA: with filters for dvipa only.

Example 3-38 NETSTAT VIPADYN,DVIPA on system SC31

```
RO SC30,D TCPIP,TCPIPA,N,VIPADYN,DVIPA
   RESPONSE=SC30
    EZD0101I NETSTAT CS V1R13 TCPIPA 335
    DYNAMIC VIPA:
      IPADDR/PREFIXLEN: 10.1.8.21/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
      IPADDR/PREFIXLEN: 10.1.8.22/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
      IPADDR/PREFIXLEN: 10.1.8.23/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
      IPADDR/PREFIXLEN: 10.1.8.25/24
        STATUS:  BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
        ACTTIME: 10/01/2010 19:18:29
    4 OF 4 RECORDS DISPLAYED
    END OF THE REPORT
```

Example 3-39 shows SC31 Netstat output for VIPADYN,VIPAROUTE, with filters for viparoute only.

Example 3-39 NETSTAT VIPADYN,VIPAROUTE on system SC31

```
RO SC30,D TCPIP,TCPIPA,N,VIPADYN,VIPAROUTE
   RESPONSE=SC30
   EZD0101I NETSTAT CS V1R13 TCPIPA 387
    VIPA ROUTE:
      DESTXCF: 10.1.7.21
        TARGETIP: 10.1.1.20
        RTSTATUS: ACTIVE
    1 OF 1 RECORDS DISPLAYED
```

3.4 FTP client using batch

This section describes running the FTP client as a batch job. It has the following topics:

- Description of FTP client using batch
- Configuration of FTP client using batch
- Activation and verification of FTP client batch job

3.4.1 Description of FTP client using batch

In addition to the dependencies discussed in 3.2, “Basic FTP without security” on page 148, an FTP batch job requires JCL containing the FTP commands to be executed.

Submitting batch FTP jobs allows non-interactive FTP sessions that run in the background.

In general, batch FTP does not handle transient failures. For example, if there is a network problem, the FTP batch job will simply fail. If the network problem is resolved, the batch job

must be resubmitted to complete the file transfer. However, capabilities can handle transient failures that are caused because a needed data set is held by another job or process.

You can use the DSWAITTIME statement to command FTP to retry access to data sets that are not available because of other users. The value specified for DSWAITTIME is the number of minutes that FTP tries to access an MVS data set that cannot be obtained because another job or process was holding the data set. FTP then tries to access the data set approximately every minute for the number of minutes specified in the DSWAITTIME statement. For example, use the following code to set the data set wait time to 10 minutes:

```
DSWAITTIME 10
```

3.4.2 Configuration of FTP client using batch

Example 3-40 shows a sample batch job.

Example 3-40 Sample FTP client batch job

```
//BATFTP   JOB MSGCLASS=X,NOTIFY=&SYSUID
//FTPCLNT  EXEC PGM=FTP,PARM='(TIMEOUT 15'            1
//NETRC    DD DSN=CSO3.NETRC,DISP=SHR                 2
//SYSFTPD  DD DSN=TCPIPB.TCPPARMS(FTPCB),DISP=SHR     3
//OUTPUT   DD *
//SYSPRINT DD *
//INPUT    DD *
10.1.1.10                                            4
LS                                                   5
QUIT                                                 6
/*
```

The following items explain Example 3-40:

1. Client parameters can be passed to the client by way of the PARM= field. See *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780 for a complete list of parameters and their syntax.

2. The NETRC data set can be used to provide the user ID and password to the FTP client program, which in turn, forwards the user ID and password to the server when prompted by the server. We used the NETRC data set from which to retrieve the user ID and password so they would not be exposed in the input command stream. If you do not include the NETRC data set in the JCL, both the user ID and the password must be included in the INPUT stream immediately after the target device specification.

3. The SYSFTPD statement points to the FTP.DATA client configuration file. See *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for a complete list of statements and their settings.

4. The first entry in the INPUT command file is the IP address or DNS name of the target server to which to connect. Alternatively, the target can be specified in the PARM= field on the EXEC statement, and if it is, it should not be specified in the IINPUT command file. Here is a sample where the target is specified in the PARM= field:

```
//FTPCLNT EXEC PGM=FTP,PARM='10.1.1.10 (TIMEOUT 15'
```

5. After the target specification, you can include any FTP subcommand in the INPUT stream. See *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780 for a complete list of FTP subcommands that can be specified.

6. The last subcommand should always be either CLOSE or QUIT.

Example 3-41 shows a sample NETRC data set.

Example 3-41 Sample NETRC data set

```
******************************** Top of Data ********************
  1       2                          3    4         5     6
MACHINE 127.0.0.1                   LOGIN CSO3     PASSWORD CSO3
MACHINE WTSC30.ITSO.IBM.COM         LOGIN CSO3     PASSWORD CSO3
MACHINE WTSC31.ITSO.IBM.COM         LOGIN CSO3     PASSWORD CSO3
MACHINE WTSC32.ITSO.IBM.COM         LOGIN CSO3     PASSWORD CSO3
MACHINE WTSC33.ITSO.IBM.COM         LOGIN CSO3     PASSWORD CSO3
MACHINE 10.1.1.10                   LOGIN CSO3     PASSWORD CSO3
MACHINE 10.1.1.20                   LOGIN CSO3     PASSWORD CSO3
MACHINE 10.1.1.30                   LOGIN CSO3     PASSWORD CSO3
MACHINE 10.1.1.40                   LOGIN CSO3     PASSWORD CSO3
******************************** Bottom of Data ******************
```

The following items explain Example 3-41:

- 1, 3, 5 The words *machine*, *login*, and *password* are keywords and must be spelled as seen here. They must also be in the same order as seen here. However, they are *not* case sensitive. If you do not provide the *login* keyword and *user ID value* on a specific machine record, the client expects to find the user ID in the INPUT stream. If you do not provide the password value after the *password* keyword on a specific machine record, the client expects to find the password in the INPUT stream (that would be a bad choice to make).
- 2 The remote device can be specified with either its IP address or its DNS name, or both, as shown.
- 4 Your user ID on the remote system must follow the *login* keyword. It is case sensitive only if RACF requires case sensitive user IDs.
- 6 Your password on the remote system must follow the *password* keyword. It is case sensitive only if RACF requires case sensitive passwords.

3.4.3 Activation and verification of FTP client batch job

Submit the batch job for execution and check its output log. Example 3-42 shows the client output log from our batch job.

Example 3-42 Sample output log for FTP batch job

```
1 EZA1736I FTP (TIMEOUT 15 TCP TCPIPB
2 EZY2640I Using dd:SYSFTPD=TCPIPB.TCPPARMS(FTPCB31) for local site configuration
  EZA1450I IBM FTP CS V1R13
  EZA1466I FTP: using TCPIPB
  EZA1456I Connect to ?
  EZA1736I 10.1.1.10
3 EZA1554I Connecting to:   10.1.1.10 port: 21.
4 220-FTPDA1 IBM FTP CS V1R13 at WTSC30.ITSO.IBM.COM, 20:09:25 on 2010-10-01.
  220 Connection will close if idle for more than 5 minutes.
5 EZA1701I >>> USER CSO3
  331 Send password please.
  EZA1701I >>> PASS
6 230 CSO3 is logged on.  Working directory is "CSO3.".
7 EZA1460I Command:
```

```
8 EZA1736I LS
  EZA1701I >>> PORT 10,1,1,20,4,12
  200 Port request OK.
  EZA1701I >>> NLST
  125 List started OK.
  EZA2284I BRODCAST
  EZA2284I HFS
  EZA2284I JCL.CNTL
  EZA2284I NETRC
  250 List completed successfully.
  EZA1460I Command:
9 EZA1736I QUIT
```

In this example, the numbers correspond to the following information:

1. The FTP client program echoes the PARM= field contents.
2. The FTP.DATA file that the resolver located is listed.
3. The FTP client indicates to which IP address and port it is connecting.
4. All server reply messages start with a 3-digit number. Server replies are documented in *z/OS Communications Server: IP and SNA Codes*, SC31-8791.
5. The FTP client program has retrieved the user ID (CS03) and password from the NETRC data set and sends them to the server with the USER and PASS subcommands, respectively.
6. The server replies with message 230, indicating that it has accepted the user ID and password and has successfully logged the user in. It has set the default working directory to a high level qualifier that is equal to the user's ID (CS03.)
7. The client FTP program prompts the user for the next subcommand. The commands are being read from the INPUT stream by the client program.
8. The next command from the INPUT stream is LS, and is sent to the server by the client program. The server replies with the 125 message, the requested information, and the 250 message.
9. The last command should always be QUIT.

3.5 FTP client application program interface

z/OS Communications Server provides an FTP client application program interface (API) that you can use to control the z/OS FTP client through programs. This FTP client API is suitable for applications written in COBOL, C/C++, Assembler, PL/I, REXX, or Java.

z/OS Communications Server: IP Programmer's Guide and Reference, SC31-8787 lists additional requirements of the application.

3.5.1 FTP client API for REXX

The FTP client API allows any application to invoke directly the functionality of the FTP client that is running on z/OS with significantly improved automation capabilities for file transfer operations. REXX skills are becoming more prevalent because of the inherent ease in programming that it provides. As such, many people can develop REXX programs to accomplish complex tasks quickly and easily.

The REXX FTP Client API is supported in the following environments:

- Non-authorized TSO exec
- Authorized TSO exec
- Batch environment
- ISPF
- UNIX environment under UNIX System shell scripts

As with an FTP batch job, an application that uses the FTP client API must handle transient errors or the FTP transfer will fail.

A REXX FTP client requires a stub routine to translate between the string format that is used within REXX programs and the text or binary format that is used by the underlying callable FTP client API.

REXX FTP client API functions share a common return code format.

3.5.2 FTP client API for Java

The FTP client API for Java provides an interface to the z/OS FTP client that enables a user program written in Java to send subcommands for the client to process. The user program can also use this interface to retrieve output that includes the messages from the client, replies from the FTP server, and other data that is generated as the result of the request.

The z/OS FTP client, when started with the FTP client API for Java, operates as it does when invoked under the z/OS UNIX shell. FTP client API for Java uses the Java Native Interface (JNI) to interface with the z/OS FTP client using the C Java FTP client API.

When using the FTP client API for Java, keep in mind the following points:

- The user program can have more than one FTP client object initialized and active in a single address space.
- All requests that use the same FTP client object must be made from the same thread.
- The application must have an OMVS segment defined (or set by default).
- The interface module EZAFTPKI must be accessible to the application in the link list or in a STEPLIB or JOBLIB DD statement.
- You must include the `EZAFTP.jar` file in your class path, and the `libEZAFTP.so` file must be located in $LIBPATH for the JNI methods to be found.

  The `EZAFTP.jar` file is installed into the `/usr/include/java_classes` directory, and the `libEZAFTP.so` file is installed into the `/usr/lib` directory.

For more information about the FTP client API for Java, follow these steps to download the JavaDoc:

1. Download the JavaDoc stored in z/OS `/usr/include/java_classes/EZAFTPdoc.jar` to your workstation (use FTP to retrieve it as a binary file).
2. Extract the documentation using the Java **`jar`** utility:

   ```
   jar –xvf EZAFTPdoc.jar
   ```

   To use the **`jar`** utility, you need Java installed on your workstation.
3. Use a web browser to view the extracted `index.html` file.

3.6 FTP access to UNIX named pipes

This section provides an overview of FTP access to UNIX named pipes and examples about how to use this function and includes the following topics:

- What are UNIX named pipes
- Description of FTP access to UNIX named pipes
- FTP configuration options
- Use the z/OS FTP client to create a named pipe in the z/OS FTP server
- Supported z/OS FTP subcommands
- Storing into a named pipe

3.6.1 What are UNIX named pipes

A *UNIX named pipe* is identified by path name in the UNIX file system. You can create a UNIX named pipe from the shell or from your program. It provides a conduit for speedy movement of data from one process to another. From the shell, you can use following commands:

- `mkfifo`
- `mknod`

From the program, you can use following commands:

- `mkfifo()`
- `mknod()`

Note: The Portable Operating System Interface for UNIX (POSIX) standard states that **`mkfifo`** is preferred to **`mknod`**.

A named pipe is visible in the z/OS UNIX file system, as shown in Example 3-43.

Example 3-43 The output of the z/OS UNIX ls command

```
CS05 @ SC31:/work>mkfifo test.fifo
CS05 @ SC31:/work>ls -la
total 40
drwxr-xr-x   2 SYSPROG  SYS1        8192 Aug 28 08:35 .
drwxr-xr-x  23 SYSPROG  SYS1        8192 Aug 27 19:13 ..
prw-r--r--   1 SYSPROG  SYS1           0 Aug 28 08:35 test.fifo
-rw-r--r--   1 SYSPROG  SYS1          20 Aug 28 08:34 test.txt
CS05 @ SC31:/work>
```

In this example, `test.fifo` file is a `named_pipe`, and `test.txt` is an ordinary file. An ordinary file is called a *regular* file or *normal* file in UNIX nomenclature. So, we use regular file to refer to an ordinary file in the z/OS UNIX file system. You can identify which file is a named pipe by viewing the file permissions. When the first character of the permissions is `p` as shown for the file `test.info`, the file is the path name of a named pipe. You can list, rename, move, delete, and **`chmod`** named pipes using UNIX shell command just as you can regular files.

Note: The size of the named pipe is zero, which is always the case, even when processes are actively writing to the named pipe. However, a size zero is not sufficient to identify a named pipe because a regular file can also be size zero.

Unlike a regular file, a named pipe is opened for writing and for reading at the same time. Any data that a process writes to a named pipe is not stored in the file system. The contents of a pipe reside in a finite buffer of volatile storage.

3.6.2 Description of FTP access to UNIX named pipes

FTP access to UNIX named pipes allows you to transfer files to and from z/OS UNIX System Services named pipes. When regular files are transferred by FTP, after files are stored in z/OS, then the files are processed by z/OS applications as needed. If the application supports reading from a z/OS UNIX named pipe, the transfer from the remote site can overlap processing by that z/OS application.

In other words, you have an unbroken pipe from the remote site that goes through a TCP connection and continues all the way into the post-processing application without further store and forward delays, as illustrated in Figure 3-4. This figure is based on the IBM DB2® batch load utility.

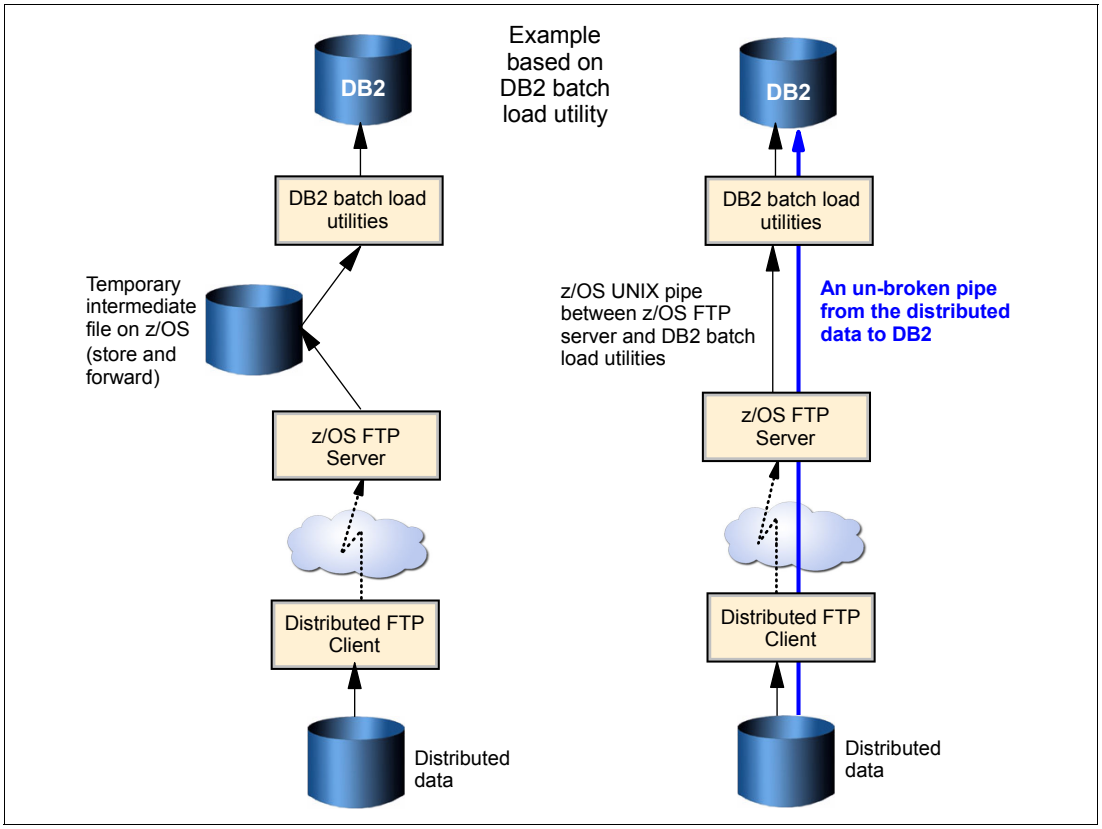


Figure 3-4 How FTP access to UNIX named pipes works

Using FTP access to UNIX named pipes provides the following benefits:

- An I/O transfer to a named pipe is faster than an I/O transfer to a regular file.
- Applications can run simultaneously with file transfers.

In addition, using FTP access to UNIX named pipes has the following limitations:

- Anonymous users cannot create, rename, delete, read from, or write to named pipes in the FTP server z/OS UNIX System Services file system.
- You can append to but not replace the contents of a named pipe.
- The operation system provides no serialization for named pipes. Multiple processes can read from or write to a named pipe.

3.6.3 FTP configuration options

The following configuration options support storing into and sending from named pipes. These options are supported for both the FTP client and the FTP server. You can code statements in FTP.DATA, and you can change these values with the FTP subcommands. See *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780 for details about the FTP subcommands.

- UNIXFILETYPE
- FIFOOPENTIME
- FIFOIOTIME

UNIXFILETYPE statement in FTP.DATA

You can use the UNIXFILETYPE statement in FTP.DATA to indicate whether to treat z/OS UNIX System Services files as regular files or as UNIX named pipes. When you code the UNIXFILETYPE FILE statement in FTP.DATA, you can create, store into, or send data from regular files, but you cannot use z/OS FTP to store into or send from named pipes. When you code UNIXFILETYPE FIFO, you can store into or send data from named pipes, but not regular files.

FIFOOPENTIME statement in FTP.DATA

You can use the FIFOOPENTIME statement to define the length of time that FTP waits after attempting to open a named pipe before reporting an error. You cannot open a named pipe for writing until another process opens the named pipe for reading, and you cannot open a named pipe for reading until another process opens the named pipe for writing. When you code FIFOOPENTIME 60 statement in FTP.DATA, z/OS FTP waits up to 60 seconds for another process to open the named pipe. If no other process opens the named pipe in FIFOOPENTIME seconds, z/OS FTP fails the file transfer.

FIFOIOTIME statement in FTP.DATA

You can use the FIFOIOTIME statement to set a time-out for reading and writing to a named pipe. If z/OS FTP or any other process tries to read from a named pipe that is empty, the read will block until the other process writes to it. It is also true that if z/OS FTP or any other process tries to write to a named pipe that is filled up with data, the write will block until the other process reads from it. When you code FIFOIOTIME 20 in FTP.DATA, z/OS FTP waits up to 20 seconds for the read or the write to complete. If the read or the write does not complete within FIFOIOTIME seconds, FTP will fail the transfer.

The locsite and site subcommands

You can change the client's configured values for UNIXFILETYPE, FIFOOPENTIME, and FIFOIOTIME after you start the z/OS FTP client by using the following locsite subcommands.

- `locsite unixfiletype={file|fifo}`
- `locsite fifoopentime=`*`seconds`*
- `locsite fifoiotime=`*`seconds`*

You can also change the server's configured values for UNIXFILETYPE, FIFOOPENTIME, and FIFOIOTIME after you log in to the z/OS FTP server with the z/OS FTP client using the following site subcommands.

- `site unixfiletype={file|fifo}`
- `site fifoopentime=`*`seconds`*
- `site fifoiotime=`*`seconds`*

The locstat and stat subcommands

You can display the client's UNIXFILETYPE, FIFOOPENTIME, and FIFOIOTIME settings with the **locstat** subcommand, as shown in Example 3-44.

Example 3-44 The locstat command reply

```
Command:
locstat
Trace: FALSE, Send Port: FALSE
(...lines deleted...)
local site variable SEQNUMSUPPORT is set to FALSE
local site variable UNIXFILETYPE is set to FILE
local site variable FIFOIOTIME is set to 20
local site variable FIFOOPENTIME is set to 60
Authentication mechanism: None
(...lines deleted...)
```

You can also display the server's UNIXFILETYPE, FIFOOPENTIME, and FIFOIOTIME settings with the **stat** subcommand when you log in to the z/OS FTP server with the z/OS FTP client, as shown in Example 3-45.

Example 3-45 The stat command reply

```
Command:
stat
>>> STAT
211-Server FTP talking to host ::ffff:10.1.2.21, port 1053
(...lines deleted...)
211-Timer DSWAITTIME is set to 0
211-Timer FIFOOPENTIME is set to 60
211-Timer FIFOIOTIME is set to 20
211-VCOUNT is 59
(...lines deleted...)
211-Server site variable UNICODEFILESYSTEMBOM is set to ASIS
211-Server site variable UNIXFILETYPE is set to FIFO
211-DBSUB is set to FALSE
(...lines deleted...)
```

You can use the z/OS FTP **stat** subcommand with a parameter to display the configured value of UNIXFILETYPE, FIFOOPENTIME, and FIFOIOTIME, as shown in Example 3-46.

Example 3-46 The stat command with a parameter reply

```
Command:
stat (unixfiletype
>>> XSTA (unixfiletype
211-Server site variable UNIXFILETYPE is set to FIFO
211 *** end of status ***
Command:
stat (fifoopentime
>>> XSTA (fifoopentime
211-Timer FIFOOPENTIME is set to 60
211 *** end of status ***
Command:
stat (fifoiotime
>>> XSTA (fifoiotime
211-Timer FIFOIOTIME is set to 20
211 *** end of status ***
Command:
```

3.6.4 Use the z/OS FTP client to create a named pipe in the z/OS FTP server

Using the z/OS FTP clients, you can create a named pipe in the z/OS FTP servers with the **mkfifo** subcommand.

The mkfifo subcommand

You can use the **mkfifo** subcommand to create a named pipe on a server when you log in to the z/OS FTP server with the z/OS FTP client, as shown in Example 3-47. You can specify absolute or relative path name.

Example 3-47 Creating a named pipe

```
Command:
mkfifo my.fifo
>>> XFIF my.fifo
257 named pipe /work/my.fifo created
Command:
ls -la
>>> PORT 10,1,2,21,4,34
200 Port request OK.
>>> NLST -la
125 List started OK
total 32
drwxrwxrwx   2 SYSPROG  SYS1        8192 Sep  4 13:20 .
drwxr-xr-x  24 SYSPROG  SYS1        8192 Sep  4 13:13 ..
prwxr-x---   1 CS05     SYS1           0 Sep  4 13:20 my.fifo
250 List completed successfully.
Command:
```

When you issue the **mkfifo** subcommand from the FTP client, the client sends an XFIF command to the server. The server replies with reply code 257 to indicate it created the named pipe successfully.

The site umask subcommand

Permissions assigned to a named pipe are affected by an FTP server configured UMASK value. The default UMASK value of z/OS FTP is 027 that causes regular files and named pipes to be created with permissions `rwxr-x---`. You can use the **site** subcommand to set the server UMASK value for the current session when you log in to the z/OS FTP server with the z/OS FTP client, as shown in Example 3-48.

Example 3-48 Configuring UMASK and creating a named pipe

```
Command:
site umask 000
>>> SITE umask 000
200 SITE command was accepted
Command:
mkfifo new.fifo
>>> XFIF new.fifo
257 named pipe /work/new.fifo created
Command:
ls -la
>>> PORT 10,1,2,21,4,10
200 Port request OK.
>>> NLST -la
125 List started OK
total 32
drwxrwxrwx   2 SYSPROG  SYS1        8192 Sep  8 16:15 .
```

```
drwxr-xr-x  24 SYSPROG  SYS1        8192 Sep  4 18:02 ..
prwxr-x---   1 CSO5     SYS1           0 Sep  8 15:46 my.fifo
prwxrwxrwx   1 CSO5     SYS1           0 Sep  8 16:15 new.fifo
250 List completed successfully.
Command:
```

This example uses a UMASK value of 000; therefore, `new.fifo` is created with permissions `rwxrwxrwx`.

3.6.5 Supported z/OS FTP subcommands

You can specify a named pipe as an argument on the following subcommands.

- append
- delete
- dir
- get
- locsite chmod
- ls
- mdelete
- mget
- mput
- put
- rename
- site chmod

Note: No FTP client will prevent you from specifying a named pipe. It always depends on the server to determine whether it supports access to a named pipe.

3.6.6 Storing into a named pipe

Before you start a transfer to or from a named pipe, you must configure the following options on the named pipe host:

- FILETYPE=SEQ (the default value)
- UNIXFILETYPE=FIFO

You must start an application that can read from the named pipe, and it must open the named pipe. Example 3-49 demonstrates storing data into a named pipe at the FTP client.

Example 3-49 Storing into a named pipe in the client file system

```
EZA1460I Command:
locsite unixfiletype=fifo
EZA1460I Command:
get /etc/hosts /work/my.fifo
EZA1733I waiting up to 60 seconds for read process to open /work/my.fifo
EZA1701I >>> PORT 10,1,2,21,4,29
200 Port request OK.
EZA1701I >>> RETR /etc/hosts
125 Sending data set /etc/hosts
250 Transfer completed successfully.
EZA1617I 1338 bytes transferred in 0.005 seconds.  Transfer rate 267.60 Kbytes/sec.
EZA1460I Command:
```

In this example, first, the FTP client user uses the **locsite** subcommand to set UNIXFILETYPE=FIFO to tell FTP to treat z/OS UNIX files as named pipes. Second, the FTP client user issues the **get** subcommand, specifying a named pipe as the local file. The client issues message EZA1733I, indicating that it is waiting for the pipe reader to open the named pipe /work/my.fifo. When the FTP client is able to open the file for writing, it sends the RETR command to the server to start the file transfer.

Note: This example demonstrates storing data into an existing named pipe at the FTP client. The FTP client creates a named pipe during get processing if it does not exist, just as it creates a regular file that does not exist. However, because the client and the pipe reader must start at the same time, you might prefer to create your named pipes before storing a file transfer.

Example 3-50 demonstrates storing data into a named pipe at the FTP server.

Example 3-50 Storing into a named pipe in the server file system

```
EZA1460I Command:
site unixfiletype=fifo
EZA1701I >>> SITE unixfiletype=fifo
200 SITE command was accepted
EZA1460I Command:
put /etc/hosts /work/my.fifo
EZA1701I >>> PORT 10,1,2,21,4,31
200 Port request OK.
EZA1701I >>> STOR /work/my.fifo
125 Appending to named pipe /work/my.fifo
250 Transfer completed successfully.
EZA1617I 1096 bytes transferred in 0.005 seconds.  Transfer rate 219.20
Kbytes/sec.
EZA1460I Command:
```

In this example, the FTP client user sets the UNIXFILETYPE to FIFO on the FTP server host to indicate that the server should treat z/OS UNIX files as a named pipe. Second, the FTP client user issues the **put** subcommand to store a local file local file /etc/hosts into a remote named pipe. The FTP client sends a STOR subcommand to the server, specifying the z/OS UNIX path name. The server is able to open the named pipe for writing right away because the named pipe reader is already active.

Note: In this example, the server issues the reply 125 Appending to named pipe /work/my.fifo. Notice the server reply says it is appending to the named pipe rather than storing into the named pipe, because all writes to a named pipe are appends. It is not possible to overlay the contents of a named pipe with new data, whether you are using FTP or another process to store into the named pipe

3.7 FTP large data set access

This section provides an overview of FTP support for extended address volumes and large format data sets, and gives an example of how to use this function.

3.7.1 The extended address volume

This section provides general information about the extended address volume and FTP support for data sets.

General information

The term *extended address volume* (EAV) refers to a volume of more than 65,520 cylinders. The support for EAV is included in the DFSMS product and requires customization by the system programmer.

Figure 3-5 shows that cylinders up to, but not including, cylinder 65,536 are in the base addressing space of the EAV. Cylinders starting with cylinder 65,536 are in the Extended Addressing Space (EAS) of the EAV.

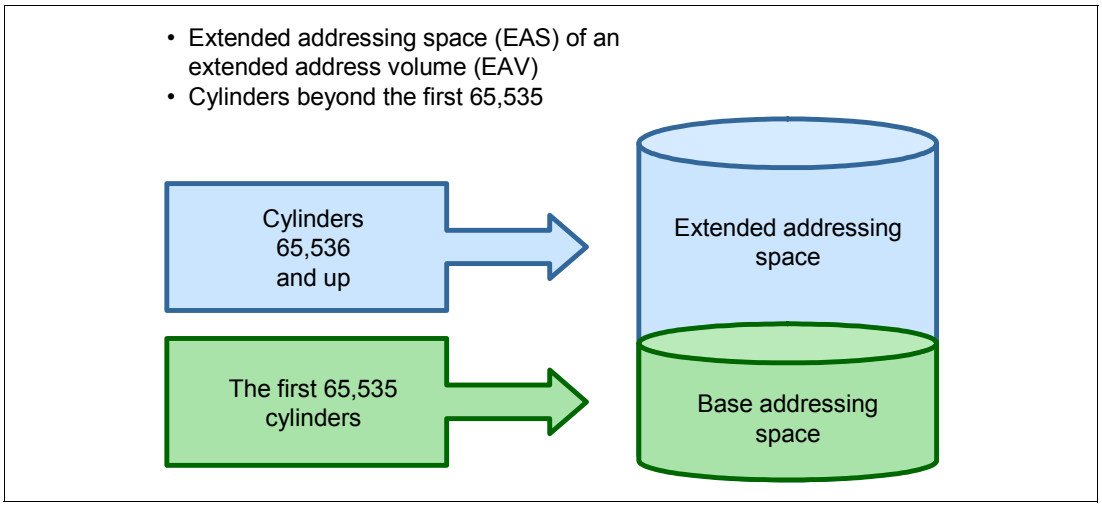


Figure 3-5 Extended addressing space (EAS) of an extended address volume (EAV)

An *EAS-eligible* data set can use cylinder-managed space. It is the only type of data set that is allowed to reside in the EAS of an EAV. However, z/OS might put an EAS-eligible data set in the base addressing space. Thus, the entirety of an EAV is available to an EAS-eligible data set.

Note: Applications that deal with volume capacity do not automatically support EAVs. An example is an application that accesses volumes to determine the amount of free space on the volume. You need to modify these applications to support EAVs.

FTP support for EAS-eligible data sets

FTP supports the following types of data sets:

- Physical sequential data sets in extended format, large format, and base format
- Partitioned data sets and extended partitioned data sets

These data sets may be either SMS-managed or not.

Note: You cannot allocate z/OS UNIX files as EAS-eligible files.

Supported FTP functions for EAS-eligible data set are as follows:

- FTP transfer from EAS-eligible data set
- FTP transfer to EAS-eligible data set

  Three approaches are available for FTP transfer to an EAS-eligible data set:

  - Allocate an EAS-eligible data set with FTP subcommand SITE EATTR/LOCSITE EATTR.
  - FTP transfer to a data set that is already allocated as EAS-eligible
  - FTP transfer to a data set in an EAV volume that is defined in an SMS data class with default attribute of EATTR=OPT. Use FTP subcommand DATACLASS to specify the SMS data class.
- Support for FTP configuration data set as EAS-eligible

  The following FTP configuration files can be EAS-eligible data sets:

  - ANONYMOUSLOGINMSG
  - ANONYMOUSMVSINFO
  - BANNER
  - LOGINMSG
  - MVSINFO
  - NETRC
  - SOCKSCONFIGFILE
  - FTP.DATA

  Note: The TSO HOMETEST command cannot process FTP.DATA when it is an EAS-eligible data set.

- Rename, delete, and list the EAS-eligible data sets
- Display the space statistics of EAV with SITE QDISK/LOCSITE QDISK FTP subcommand

3.7.2 FTP support for large format data set

Physical sequential large format data sets can be as large as 16,777,215 tracks per volume (up to 59 volumes), where the physical sequential basic format supports 65,535 tracks per volume. Physical sequential large format data sets need not be large, the minimum size is one track.

FTP support for physical sequential large format data sets is as follows:

- Transfer to and from physical sequential large format data sets.

  You may configure FTP to allocate new physical sequential data sets as basic format or large format by specifying FTP subcommand SITE DSNTYPE/LOCSITE DSNTYPE.
- FTP configuration file can be a physical sequential large format data set.
- Block mode restart of interrupted file transfer (specify FTP subcommand RESTART) is supported.
- Rename, delete, list the physical sequential large format data set.

3.7.3 Example of EAS-eligible data set allocation for FTP transfer

Example 3-51 demonstrates how to allocate an EAS-eligible partitioned data set (PDS) for FTP transfer. The FTP client issues a SITE FTP subcommand to set EATTR to OPT, the server volume to an EAV volume, and PDSTYPE to PDS (use QUOTE with SITE subcommand if you use a non-z/OS FTP client). EATTR OPT allows the system to allocate an EAS-eligible data set if the volume on which the data set resides is an EAV. To ensure the allocation to an EAV, specify the name of the EAV with the VOLUME parameter. Then, the FTP client issues a MKDIR FTP subcommand to create a new EAS-eligible PDS.

Example 3-51 Allocating an EAS-eligible data set from an FTP client

```
quote site eattr=opt volume=EAVVOL pdstype=pds
200 SITE command was accepted
quote stat
...
211-Data sets will be allocated on EAVVOL.
...
211-Server site variable EATTR is set to OPT
211-Server site variable DSNTYPE is set to SYSTEM
mkdir 'user1.eas.pds'
257 "'USER1.EAS.PDS'" created.
```

3.8 Miscellaneous configuration settings of FTP

Other useful functions of FTP are available that we do not cover in this book. However, you should be aware of the capabilities. In this section, we discuss the following topics:

- A single generic FTP server in a multiple stack z/OS image
- FTP network management interface with SMF

3.8.1 A single generic FTP server in a multiple stack z/OS image

In this section we discuss starting a single FTP server to handle connections through multiple stacks on the same LPAR. The multiple stacks per z/OS image is referred to as a CINET environment.

Dependencies of a generic FTP server

Multiple stacks on the LPAR must be configured and active.

Advantages of a generic FTP server

If you need two or more stacks configured differently but only one FTP server is required, then the FTP server can be set up as a generic listener. This configuration enables the FTP server to listen for connections on all stacks. For example, you could have one stack with a firewall and policies active which processes connections from an external network, and one stack without a firewall and policies which processes connections from the internal network. If one single FTP configuration is capable of meeting the needs of all stacks, then this configuration has an additional advantage: only one FTP instance needs to be configured.

Considerations for using a generic FTP server

Additional stacks require additional resources in terms of CPU load and memory usage. Additional system definitions are required to accommodate multiple stacks per system image

(known as a CINET environment). Running multiple stacks per system image is not recommended. The requirement for multiple stacks is best accomplished using multiple LPARs and sysplexes.

3.8.2 FTP network management interface with SMF

System Management Facility (SMF) is a component of z/OS. It is used to help monitor z/OS systems by capturing and recording events that occur. After these events are recorded, reports can be generated either by user-written programs that format the data into a readable format or by real time network monitors for display. These recorded events are referred to as SMF Records, and are stored in one of two places:

- SMF data sets: VSAM data sets, typically named SYS1.MANx, SYS1.MANy, and so on
- z/OS dataspaces

SMF data sets must have one data set actively recording data.

The SMF records for FTP are as follows:

- Three SMF records provide session information:
  - FTP client login failure
  - FTP client session
  - FTP server session
- Five records provide more detailed information:
  - FTP client transfer completion record
  - FTP server transfer completion record
  - FTP server login failure
  - FTP client transfer initialization
  - FTP server transfer initialization

Advantages

The SMF data can be accessed in two ways:

- Through normal batch processing of SMF data
- Through a real-time Network Management interface

For normal SMF data, you must configure profile.tcpip and ftp.data to specify that SMF records are to be created when certain events occur.

- Code the SMF statement SMFCONFIG TYPE119 FTPCLIENT in the `profile.tcpip` file.
- Code the SMF statements in the FTP server's `ftp.data` file as shown in Example 3-52.

Example 3-52 Definitions in the ftp.data file

```
SMF      STD       ; SMF type 118
SMF      TYPE119   ; SMF type 119
```

The real-time network management data is enabled by including a NETMONITOR SMFSERVICE statement in profile.tcpip.

3.9 Problem determination for FTP

In case of a server problem, first check the console and syslogd for error messages. If no problem can be identified from messages, enable the FTP TRACE by specifying the keyword TRACE on the first line of the FTPD catalogued procedure on the PARMS parameter.

Also see *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

3.10 Additional information sources for FTP

See the following sources for additional information:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- FTP is defined by RFC 959

Tip: For descriptions of security considerations that affect specific servers or components, see "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

4

Simple Network Management Protocol

Simple Network Management Protocol (SNMP) enables monitoring and management of the devices and computers participating in a TCP/IP network. This chapter focuses on the SNMP services that are available in the z/OS Communications Server, and complements the product publications with practical implementation scenarios that can be useful in your environment.

A Management Information Base (MIB) provides the core definition of all network-managed resources. Managed data is defined in IETF standards, and in proprietary instances the data are called *MIB objects* or *MIB variables*. MIB-II is the recommended Internet standard format for managed information. Its current specification can be found in RFC 1213, and RFC2011 through RFC2013 (Management Information Base for Network Management of TCP/IP-based internetworks: MIB-II).

This chapter discusses the following SNMP topics.

Section	Topic
4.1, “Conceptual overview of SNMP” on page 206	The basic concepts of SNMP network management.
4.2, “z/OS SNMP agent” on page 210	SNMP Agent configuration setup examples and verification procedures.
4.3, “z/OS SNMP subagents” on page 220	SNMP Subagent configuration setup examples and verification procedures.
4.4, “z/OS SNMP client command” on page 226	SNMP client command configuration setup examples and usage procedures.
4.5, “Problem determination for the SNMP facilities” on page 235	Problem determination tools and commands for z/OS SNMP facilities.
4.6, “Additional information sources for SNMP” on page 236	References to additional documentation for z/OS SNMP facilities.

4.1 Conceptual overview of SNMP

As illustrated in Figure 4-1, SNMP is one of the standard applications provided with the z/OS Communications Server. Both the SNMP agent and the SNMP command send and receive message packets through z/OS UNIX Sockets, using the Assembler Callable API and C-Sockets. They make use of the Logical File System to pass the packets in and out of the Physical File System on to the layers of the stack.

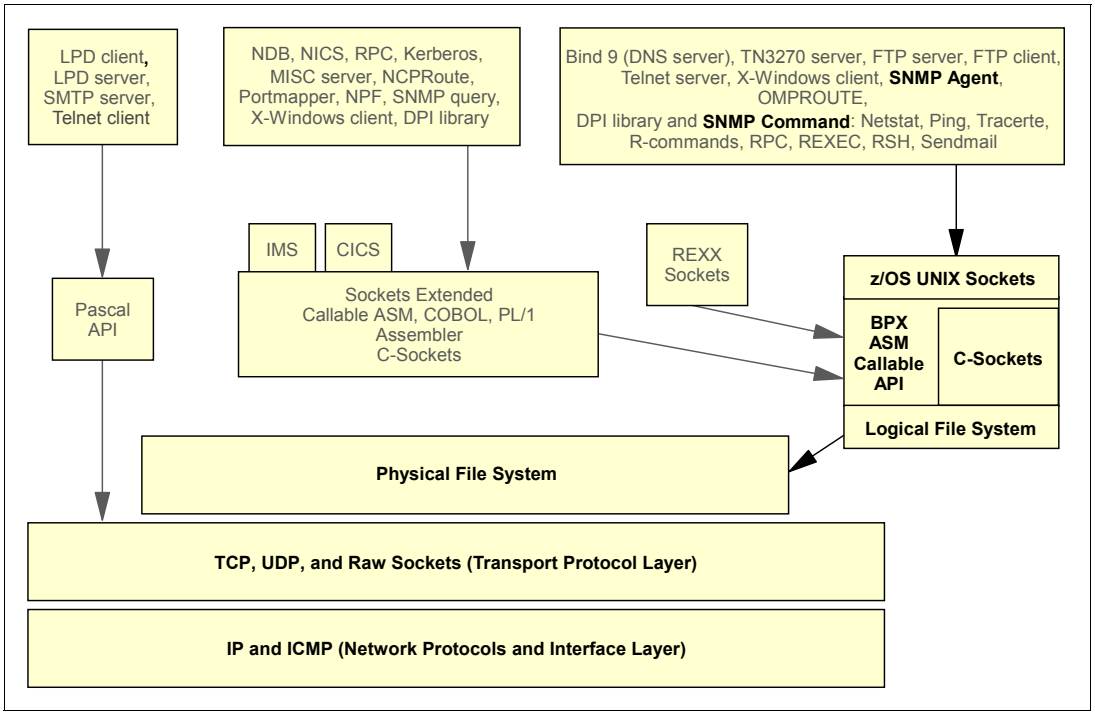


Figure 4-1 z/OS SNMP management services

As IP networks have grown in size and complexity, so has the need for managing them. Management packages are becoming more sophisticated and are now requiring the availability of an SNMP agent. The z/OS platform provides the robust support and high availability that is required by these packages.

SNMP is probably the most widely used application for managing elements in a TCP/IP network. For most vendors, it has become an integral part of their packaged network management offerings. The versatility of the z/OS platform enables it to support both SNMP managers and SNMP agents for collecting, monitoring, and reporting network statistics. A number of the applications shipped with the z/OS Communications Server have their own subagent that can be enabled to register with an SNMP agent on the same system image.

This section discusses the following topics:

- What is SNMP
- How does SNMP work
- How can SNMP be applied

4.1.1 What is SNMP

SNMP is an Internet standard protocol. Its current specification can be found in RFC 1157 Simple Network Management Protocol (SNMP). SNMP makes use of managers, agents, and

subagents. The subagents are closely associated with managed elements that are able to access specific device status information. The agents provide interfaces on behalf of subagents to management packages that want to retrieve that information.

The SNMP framework enables network administrators to address various networking management related issues. Because these operations are done using the SNMP protocol, a network administrator can reside anywhere in the IP network, and no longer has to log into the target systems to maintain network nodes.

The framework of version 2 of the Simple Network Management Protocol (SNMPv2) was published in April 1993 and consists of 12 RFCs, the first being RFC 1441 (which is an introduction). In August 1993, all 12 RFCs became proposed standards with the status *elective*. SNMPv3 is described in RFCs 2570 through 2573 and RFCs 3411 through 3415. SNMPv3 is an extension to the existing SNMP architecture.

The view-based access control model supported in SNMPv3 allows granular access control for MIB objects with either the user-based or community-based security models.

SNMPv3 also enables dynamic changes to the SNMP agent configuration. The SNMPv3 architecture is modularized so that portions of it can be enhanced over time without requiring the entire architecture to be replaced.

4.1.2 How does SNMP work

A Network Management Station (NMS) requests an item of information from the agent to which the NMS has an IP connection. The agent forwards that request to an appropriate subagent that has registered itself as supporting that type of information. The subagent prepares a response and sends the information back to the agent. The agent forwards the response back to the NMS. The NMS is not aware of how the agent acquires the information.

The SNMP protocol implementation is illustrated in Figure 4-2.

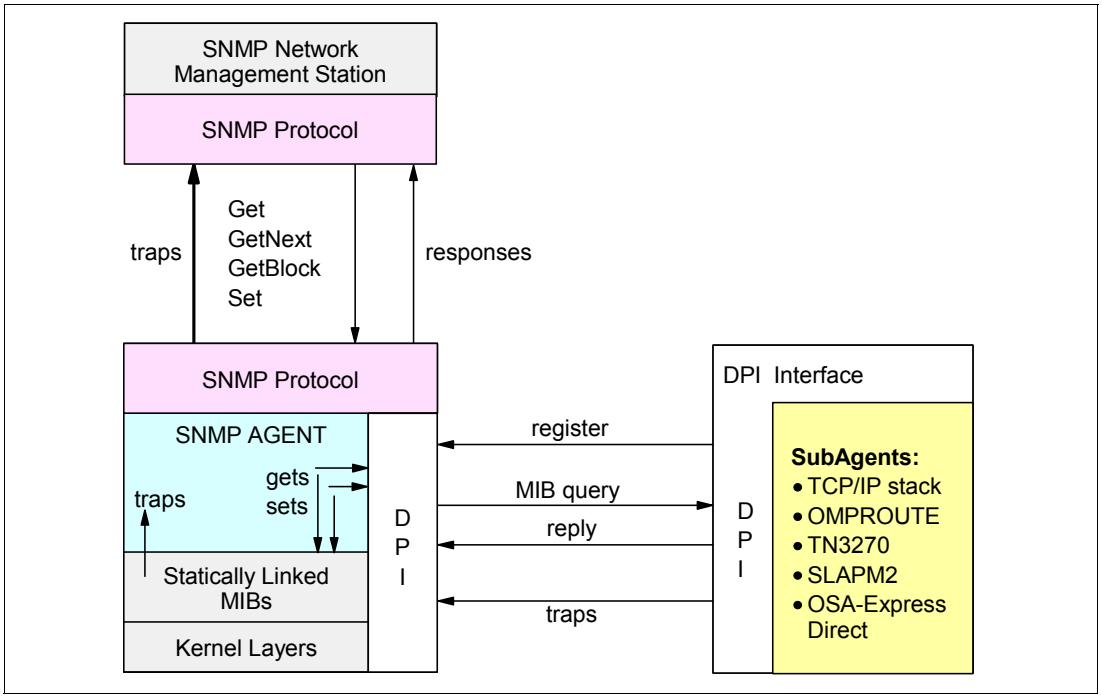


Figure 4-2 SNMP agent/subagent relationship to a Network Management Station (manager)

The SNMP Distributed Programming Interface (IBM DPI) allows a process to register the existence of an MIB variable with the SNMP agent. When requests for the variable are received by the SNMP agent, it passes the query on to the process acting as a subagent. This subagent then returns an appropriate answer to the SNMP agent. The SNMP agent packages an SNMP response packet and sends the answer back to the remote network management station that initiated the request. The network management stations have no knowledge that the SNMP agent calls on other processes to obtain an answer.

- The SNMP manager (NMS) communicates with the SNMP agent through the SNMP protocol.
- The SNMP agent communicates with the Management Information Base that represents the items of information available.
- An SNMP subagent, running as a separate process, can set up a connection with the agent. The subagent has an option to communicate with the SNMP agent through UDP or TCP sockets, or even through other mechanisms.
- After the connection is established, the subagent registers one or more of its MIBs with the SNMP agent. Multiple subagents can register with the agent concurrently. In our environment, those subagents could be from TN3270, OMPROUTE, the TCP/IP stack, Infoprint Server, and others.
- The SNMP agent receives request packets from the SNMP manager (NMS). If the packet contains a request for an object in an MIB registered by a subagent, it sends a corresponding request in a Distributed Programming Interface (DPI) packet to the subagent.
- The SNMP agent then encodes a reply into an SNMP packet and sends it back to the requesting SNMP manager.

To understand the relationship between the NMS, the agent, and the subagent, an understanding of the Distributed Programming Interface (DPI) is necessary. This is a special-purpose programming interface that you can use if you want to implement Management Information Base (MIB) variables. In an z/OS Communications Server SNMP environment the MIB variables are defined in the MIBS.DATA data set. If you want to add, replace, or delete MIB variables, you can develop an SNMP subagent program that uses the DPI programming interface to interact with the SNMP agent address space (OSNMPD) to perform such functions. For details about using the DPI interface, see *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787.

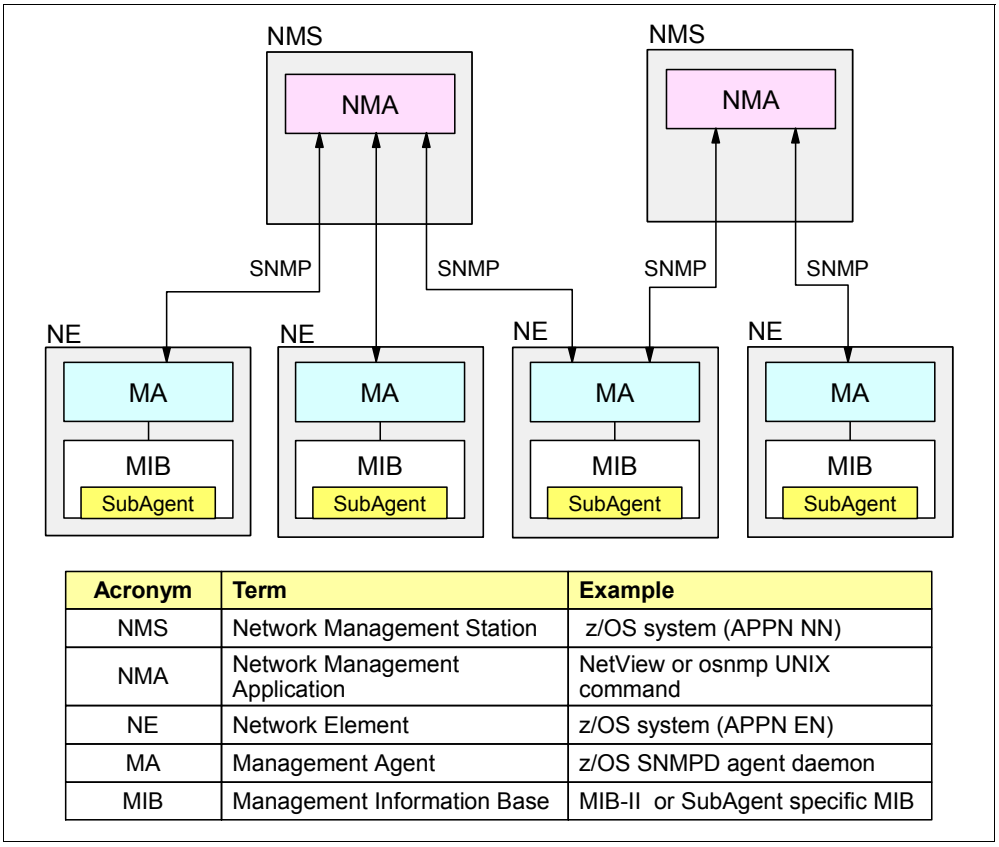
Several subagents and their specific MIBs delivered with the z/OS Communications Server are as follows:

- TCP/IP stack subagent with /usr/lpp/tcpip/samples/mvstcpip.mi2
- OMPROUTE subagent
- TN3270E Telnet server subagent with /usr/lpp/tcpip/samples/mvstn3270.mi2
- NSLAPM2 V2 subagent for policy agent with usr/lpp/tcpip/samples/slapm2.mi2
- OSA-Express Direct subagent supports data for OSA-Express features

The SNMP DPI V2.0 protocol specified in RFC1592 provides the ability to connect agents through AF_UNIX or AF_INET sockets. The list of SNMP RFCs is large, and it is out of the scope of this book to discuss them. For complete details about the standards and how they relate to each other, see *TCP/IP Tutorial and Technical Overview*, GG24-3376.

4.1.3 How can SNMP be applied

Figure 4-3 illustrates the generic implementation of SNMP in a common industry standard design. It shows the management component layers and the roles they play in network management. The IBM z/OS implementation does not use all of the terminology defined in the diagram; however, the diagram and its following explanation will assist in understanding how SNMP is generally implemented.



Acronym	Term	Example
NMS	Network Management Station	z/OS system (APPN NN)
NMA	Network Management Application	NetView or osnmp UNIX command
NE	Network Element	z/OS system (APPN EN)
MA	Management Agent	z/OS SNMPD agent daemon
MIB	Management Information Base	MIB-II or SubAgent specific MIB

Figure 4-3 SNMP Network Management roles

A host platform that runs a management package is a Network Management Station, and the package is the Network Management Application (or manager). The manager communicates with an agent using the SNMP protocol. The agent communicates with subagents to retrieve information that satisfies requests from the manager. The protocol used between an agent and a subagent can be DPI/SMUX/AgentX or any proprietary protocol. The z/OS Agent uses the DPI interface. A host platform that runs the agent and possibly the subagent is the network element.

Use caution when you are working with SNMPv3 environments that enable authentication and encryption. SNMPv3 notifications are encrypted and use a unique security key which is created by hashing user ID and engine ID. z/OS Communications Server provides a configurable engine ID. See *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787 for information about how to specify the engine ID parameter in your SNMP manager API configuration file.

We discuss the following SNMP implementations in this chapter:

- z/OS SNMP agent
- z/OS SNMP subagents
- z/OS SNMP client command

4.2 z/OS SNMP agent

A common implementation of SNMP is to set up the SNMP agent (OSNMPD) as a started task on the z/OS platform under the native MVS environment. We discuss the configuration of OSNMPD as an agent in the following topics:

- Description of the z/OS SNMP agent
- Configuration of the z/OS SNMP agent
- Activation and verification of the z/OS SNMP agents

4.2.1 Description of the z/OS SNMP agent

In the z/OS Communications Server, the **snmp** command provides SNMP network management from the z/OS UNIX shell. The IBM Tivoli® IBM NetView® SNMP command provides network management from the NetView command line.

In the z/OS Communications Server, the SNMP agent is a z/OS UNIX application. It supports SNMPv1, SNMPv2c, and SNMPv3. SNMPv3 provides a network management framework that enables the use of user-based security in addition to, or instead of, the community-based security supported in SNMPv1 and SNMPv2c.

The z/OS SNMP agent and its relationship with the IP network is illustrated in Figure 4-4.

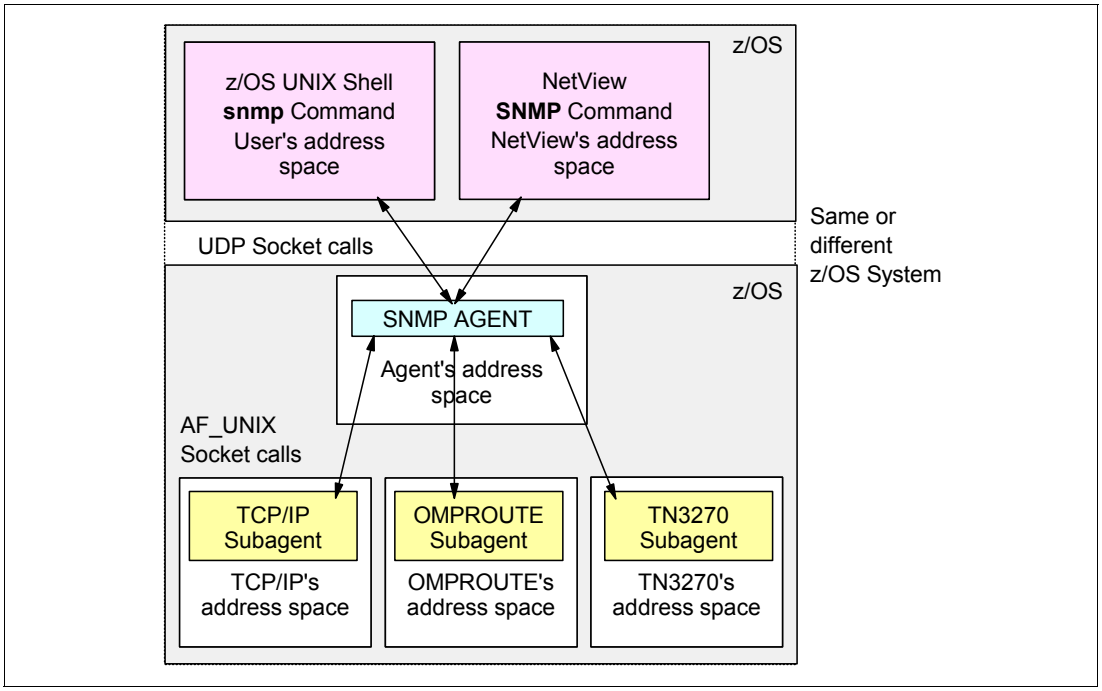


Figure 4-4 SNMP agent/subagent relationships

Dependencies of the z/OS SNMP agent

A subagent extends the set of MIB variables supported by an SNMP agent. Each subagent must be set up and configured independently. The z/OS Communications Server supports the following subagents:

- TCP/IP subagent with /usr/lpp/tcpip/samples/mvstcpip.mi2
- OMPROUTE subagent
- TN3270 Telnet subagent with /usr/lpp/tcpip/samples/mvstn3270.mi2

- Network SLAPM2 subagent with /usr/lpp/tcpip/samples/slapm2.mi2
- OSA-Express Direct subagent supports data for OSA-Express features

The TCP/IP stack must be active on the z/OS system on which the SNMP agent runs. The z/OS UNIX environment must be enabled and active. The agent is a z/OS UNIX application.

The trap forwarder task forwards traps from the SNMP agent to network management applications. It listens for traps on a port, typically 162, and forwards them to all configured managers. If you want to forward trap information to one or more managers, you must configure the trap forwarder. For setup and configuration details, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

TN3270 SNMP subagent limitation

TN3270 SNMP subagent activation requires specification of a stack name to register with the agent. Without TCPIPJOBNAME, TN3270 blocks the subagent activation request.

The TN3270 SNMP subagent can only register with one agent, and each agent can support only one TN3270 subagent. In addition to the required affinity, you must be careful to plan for one agent per TN3270 subagent, including the TN3270 subagent that might be running within the TCP/IP stack's address space. If multiple TN3270 SNMP subagents initialize to the same agent, the agent forwards all data requests to the first subagent that connected, and all other initializations are queued. If the first subagent ends, the next subagent in the queue then receives all data requests.

Considerations for using the z/OS SNMP agent

Migrating from SNMPV2 to SNMPv3 takes planning and could be a complex task. For information about migrating z/OS SNMP configuration files from SNMPv1 and SNMPv2c to SNMPv3, see the SNMP chapter in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

4.2.2 Configuration of the z/OS SNMP agent

Depending on the network management requirements of your organization, you might have to implement certain features of the SNMP agent on your z/OS platform. There are common steps to set up the SNMP agent regardless of how it is going to be used or what role it plays. We show you how to set up the basic SNMP agent in this section. The subagents are set up independently. Each has its own set of definitions and configuration file. The diagram in Figure 4-4 on page 210 is referred to in the following implementation tasks for the z/OS SNMP agent:

- Update the TCP/IP profile configuration data set for the agent
- Update the TCP/IP profile configuration data set for the query engine
- Update RACF to define the SNMP agent started task
- Customize the SNMP agent procedure JCL
- Create the environment variable file for the SNMP agent proc
- Create the MIB object configuration file for the SNMP agent
- Determine the type of security supported by the SNMP agent
- Create the community name file for the SNMP agent
- Create the trap destination file for the SNMP agent
- Configure the query engine started task (optional)
- Create the SNMPARMS file to be use by the NetView query engine
- Configure the Trap Forwarder started task and its files (optional)

Update the TCP/IP profile configuration data set for the agent

The AUTOLOG and PORT statements should be updated to indicate the action and support that the stack needs to provide for the SNMP agent. AUTOLOG indicates whether the stack should initially start the SNMP started task. PORT provides a port reservation for the port number that the SNMP server listens on. The default is port 161. The AUTOLOG and PORT statements are shown in Example 4-1.

Example 4-1 Auto starting and port reservation for the SNMPD started task

```
AUTOLOG
   SNMPDB
ENDAUTOLOG

PORT
   161 UDP SNMPDB
```

Update the TCP/IP profile configuration data set for the query engine

The SNMP agent uses port 162, by default, for sending traps to the managers specified in the SNMPTRAP.DEST or the SNMPD.CONF file. Port 162 should be reserved for the management application primarily responsible for trap processing. If you plan to use the query engine (one of the users of this is Tivoli NetView), then reserve the port as shown in Example 4-2.

Example 4-2 Auto starting and port reservation for the SNMPQE query engine

```
AUTOLOG
   SNMPQEB
ENDAUTOLOG

PORT
   162 UDP SNMPQEB
```

Update RACF to define the SNMP agent started task

Every started task must be assigned a user ID, and that user ID must be granted authority to access the required resources that support the started task. The started tasks that need to be considered here are:

- SNMPD, the agent's started task
- SNMPQE, the query engine
- TRAPFWD, the trap forwarder

The SNMP agent started task is used as an example. This discussion assumes RACF is the security subsystem being used. If another security product is used, see its manuals for equivalent set up instructions. Before SNMP can be started, security for the procedure name and its associated user ID must be defined. Review the sample file SEZAINST(EZARACF) that contains sample security statements for this effort.

The procedure name must be added to the RACF STARTED class and have a user ID associated with it as follows:

```
RDEFINE STARTED SNMP*.* STDATA(USER(SNMP))
SETROPTS RACLIST(STARTED) REFRESH
```

Coding the started task name using the wildcard format enables us to run multiple SNMP started tasks without having to define each one separately. Their names would all be spelled as SNMP*x*, where *x* is the qualifier. They can all be assigned to the same user ID.

Use an existing superuser ID, or define a superuser ID to associate with the job name by adding a user ID to RACF and altering it to superuser status as follows:

```
ADDUSER SNMP
ALTUSER SNMP OMVS(UID(0) PROGRAM ('/bin/sh') HOME('/'))
```

In this example, the user ID name is SNMP, but any name can be used. These two RACF commands can be combined into one command by putting the OMVS parameter on the ADDUSER command line. The add and alter commands are done separately in case the user ID already exists. If the add fails, the alter still succeeds.

If setting up a superuser ID is not desirable, you can instead permit the user ID to the BPX.SUPERUSER class using the following steps:

1. Add the user to RACF:

   ```
   ADDUSER OSNMP
   ```

2. Permit the user ID:

   a. Create a BPX.SUPERUSER FACILITY class profile:

      ```
      RDEFINE FACILITY BPX.SUPERUSER
      ```

   b. If this is the first class profile, activate the FACILITY class:

      ```
      SETROPTS CLASSACT(FACILITY) SETROPTS RACLIST(FACILITY)
      ```

   c. Permit the user to the class:

      ```
      ALTUSER SNMP OMVS(UID(25) PROGRAM ('/bin/sh') HOME('/'))
      PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SNMP) ACCESS(READ)
      ```

      In this example, the user ID is SNMP and the UID is 25. The UID can be any nonzero number. UID 25 was used to match the well-known SNMP port number.

   d. Refresh the FACILITY class:

      ```
      SETROPTS RACLIST(FACILITY) REFRESH
      ```

Customize the SNMP agent procedure JCL

A sample of the procedure is in `hlq.SEZAINST(OSNMPDPR)`. Customize data set names to meet installation standards. Several follow-on steps described next might require you to add one or more DD statements to this procedure. Store your updated procedure into your system proclib and make sure its name matches the name you put on the AUTOLOG and PORT statements in the TCP/IP profile configuration data set.

Our SNMP agent started task procedure is shown in Example 4-3.

Example 4-3 SNMP agent Proc JCL

```
//SNMPDB PROC PARMS='-d 0',STDENV=SNMENV&SYSCLONE.              1
//SNMPDB EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/&PARMS')
//STDENV  DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV.)  2
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*
```

The following items explain Example 4-3 on page 213:

1. The &SYSCLONE system variable is used to determine the name of the SNMPENV member that contains settings for environment variables used by SNMPD.
2. The STDENV DD statement should specify a PDS that has a variable record format (RECFM=V or VB).

Note: You can use the `_CEE_ENVFILE` environment variable in the PARM field of the JCL to point to a file that contains other environment variables. The file can be a UNIX file, a zFS, or a z/OS MVS data set.

When it is an MVS data set, the data set must be allocated with RECFM=V. RECFM=F must *not* be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in z/OS UNIX. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Create the environment variable file for the SNMP agent proc

Example 4-4 shows the two STDENV files for our two agents. The SNMP agent requires information from the files that are indicated in the environment variable file. The PDSs must have a RECFM of VB to enable proper processing of the variable settings.

Example 4-4 SNMP agent STDENV files for SC30 and SC31

```
for SC30:
BROWSE    TCPIP.SC30.STDENV(SNMENV30) - 01.01 Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA                1
RESOLVER_CONFIG=//'TCPIPA.TCPPARMS(DATAA30)'    2
OSNMPD_DATA=//'TCPIPA.TCPPARMS(SNMPD30)'        3
PW_SRC=//'TCPIPA.TCPPARMS(PWSRC30)'             4
SNMPTRAP_DEST=//'TCPIPA.TCPPARMS(TRPDST30)'     5

for SC31:
BROWSE    TCPIP.SC31.STDENV(SNMENV31) - 01.01 Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
OSNMPD_DATA=//'TCPIPB.TCPPARMS(SNMPD31)'
PW_SRC=//'TCPIPB.TCPPARMS(PWSRC31)'
SNMPTRAP_DEST=//'TCPIPB.TCPPARMS(TRPDST31)'
```

The following items explain Example 4-4:

1. _BPXK_SETIBMOPT_TRANSPORT establishes stack affinity to the indicated stack.
2. RESOLVER_CONFIG points to the TCPDATA resolver file.
3. OSNMPD_DATA points to the MIB object configuration file for the agent.
4. PW_SRC points to the community names file used by the agent
5. SNMPTRAP_DEST points to the trap destination file if needed by the agent

Create the MIB object configuration file for the SNMP agent

Supplying this information permits you to customize the values of certain MIB objects for your specific installation. A sample, containing the MIB objects to be set, can be found in the z/OS UNIX file system as file `/usr/lpp/tcpip/samples/osnmpd.data`. It can be copied to a PDS

member if you want the SNMPD procedure to use it that way. Customize the settings to match your environment. Example 4-5 shows a portion of the osnmpd_data file.

Example 4-5 OSNMPD_DATA file TCPIPB.TCPPARMS(SNMPD31)

```
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "I&O Mainframe Network Support"                          1
  sysLocation "Datacenter Mainframe Operations"                        2
  sysName "SC31 - z/OS Communications Server"                          3
# Default value of sysObjectID is equivalent to ibmTcpIpMvs
#  in the ibmAgents subtree; this is the sysObjectID representing
#  IBM z/OS Communications Server
# Changing this value is not recommended, as it is intended to allow
#  network management applications to identify this agent as the
#  z/OS Communications Server SNMP agent.  The ability to change it
#  will be disabled in a subsequent release.
# sysObjectID     "1.3.6.1.4.1.2.3.13"                                 4
  snmpEnableAuthenTraps 1
  saDefaultTimeout    10
  saMaxTimeout  700
# saAllowDuplicateIDs must be set to 1 to allow multiple DPI version 1
#  subagents
  saAllowDuplicateIDs   1
  dpiPathNameForUnixStream "/tmp/dpi_socket"
# Default value of sysServices indicates support for
#  internet, end-to-end, and application layers as
#  defined in RFC 1907.
  sysServices       76
```

The following items explain Example 4-5:

- You can set the values of 1, 2, and 3 to meaningful names and descriptions for your systems.
- 4 Do not code sysObjectID; it must remain commented.

Determine the type of security supported by the SNMP agent

You must choose to use either community-based security or a combination of community-based and user-based security. There are a number of issues to be considered in making the determination. See *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for assistance in making that decision.

- Community-based security

  If you intend to use community-based security (SNMPv1 and SNMPv2c), then set up the PW.SRC and SNMPTRAP.DEST files. The PW.SRC data set and the SNMPD.CONF data set are mutually exclusive. 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.

  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. To use traps, you must provide SNMPTRAP.DEST information defining a list of managers to which traps are sent. The SNMPTRAP.DEST data set and the SNMPD.CONF data set are mutually exclusive. Verify that there is no SNMPD.CONF file. If an SNMPD.CONF file is found, the SNMPTRAP.DEST file will not be used.

- Community-based and user-based security

  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. A sample SNMPD.CONF file is shipped as `/usr/lpp/tcpip/samples/snmpd.conf`.

  Note: Do not be confused by the names of the sample file system. The /snmpd.conf is used by the SNMP agent we are discussing. The /snmpv2.conf is used by the z/OS UNIX **osnmp** command when acting as a manager submitting a request to an agent, and is discussed later:

  - `/snmpd.conf` is referred to as SNMPD.CONF, and used by the agent
  - `/osnmp.conf` is referred to as OSNMP.CONF, and used by the command
  - `/snmpv2.conf` is referred to as OSNMP.CONF, and used by the command

  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 file 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.

Create the community name file for the SNMP agent

Example 4-6 shows our PWSRC file. Notice that we entered all the community names in upper and lower case to avoid case sensitive issues and *not found* conditions. The name `jos9m2ap` is the name we focus on in our scenario.

Example 4-6 SNMP PW.SRC file

```
BROWSE    TCPIPB.TCPPARMS(PWSRC31) - 01.00           Line 00000000 Col 001 080
public  0.0.0.0   0.0.0.0
NSS 10.15.97.0       255.255.255.0
nss 10.15.97.0       255.255.255.0
ral 10.20.0.0         255.255.0.0
RAL 10.20.0.0         255.255.0.0
j0s9m2ap 10.0.0.0       255.0.0.0
J0S9M2AP 10.0.0.0       255.0.0.0
```

Create the trap destination file for the SNMP agent

Example 4-7 shows our trap destination file. For each system, the destinations are the *other* systems in the Sysplex that are enabled (configured) to receive traps from this system.

Example 4-7 SNMPTRAP.DEST files for the SNMPD agent on each system

```
BROWSE    TCPIPA.TCPPARMS(TRPDST30) - 01.00          Line 00000000 Col 001 080
# IP ADDRESSES OF THE SYSTEMS WHERE TRAPS CAN BE FORWARDED TO
10.1.1.20 UDP
10.1.1.30 UDP
10.1.1.40 UDP

BROWSE    TCPIPB.TCPPARMS(TRPDST31) - 01.00          Line 00000000 Col 001 080
# IP ADDRESSES OF THE SYSTEMS WHERE TRAPS CAN BE FORWARDED TO
10.1.1.10 UDP
10.1.1.30 UDP
10.1.1.40 UDP
```

Configure the query engine started task (optional)

Update the SNMPQE cataloged procedure by copying the sample in SEZAINST(SNMPPROC) to your system PROCLIB. Specify SNMP parameters, change the data set names as required to suit your local configuration, and see Example 4-8 for a sample of the SNMPQE PROC JCL that we used.

Example 4-8 SNMPQE Proc JCL

```
//SNMPQEB PROC MODULE=SQESERV,PARMS=''
//SNMPQEB EXEC PGM=&MODULE,PARM='&PARMS',
//             REGION=0M,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZADSIL,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.),DISP=SHR
```

Create the SNMPARMS file to be use by the NetView query engine

Example 4-9 shows NetView SNMPARMS for use with our query engine. Place this member into NetView's DSIPARM data set.

Example 4-9 SNMPARMS NetView member interface to SNMPQE

```
  SNMPQE   SNMPQEB  * Started task name of SNMP Query Engine        1
  SNMPQERT 60       * Retry timer (seconds) for IUCV CONNECT
  SNMPRCNT 2        * Retry count for sending SNMP requests
  SNMPRITO 10       * Retry initial timeout (10ths of a second)
  SNMPRETO 2        * Retry backoff exponent (1=linear, 2=exponential)
  SNMPMMLL 80       * Line length for Multiline Messages 38/44
```

In this example, the SNMPQE keyword specifies the actual *started task name* of the SNMP Query Engine **1**. Do not set it to the user ID that the started task is running under.

Configure the Trap Forwarder started task and its files (optional)

The Trap Forwarder task forwards traps from the SNMP agent to network management applications. It listens for traps on a UDP port (typically 162) and forwards them to all configured managers. If you want to forward trap information to one or more managers, you must configure the Trap Forwarder. Set up the Trap Forwarder's started task JCL, its environment variable file, and its configuration file.

Note: If you are going to run both the SNMP Query Engine started task and the trap forwarder started task (both listen on UDP port 162 by default), then Trap Forwarder has to listen on a different port from 162 to avoid conflicts. An alternative is to let them both listen on port 162 but to make the Trap Forwarder to be a bind-specific listener by coding BIND on its port reservation statement and by specifying a dynamic VIPA address for the forwarder. A dynamic VIPA address used for the BIND is specified with the VIPARANGE statement. Client threads sending packets to the forwarder then specify that bind address to contact the forwarder.

Configure the Trap Forwarder started task JCL

Example 4-10 shows the Trap Forwarder procedure JCL.

Example 4-10 Trap Forwarder proc JCL, TRAPFWDB

```
BROWSE    TCPIPB.TCPPARMS(TRAPFWDB) - 01.01  Line 00000000 Col 001 080
//TRAPFWDB PROC PARMS='-d 0',STDENV=TRPENV&SYSCLONE.                    1
//TRAPFWDB EXEC PGM=EZASNTRA,REGION=0M,TIME=NOLIMIT,
//         PARM=('POSIX(ON) ALL31(ON)',
//         'ENVAR("_CEE_ENVFILE=DD:STDENV")/-p 1162 &PARMS')
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV.)          2
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*
```

The following items explain Example 4-10:

1. The &SYSCLONE system variable is used to determine the name of the TRAPENV member that contains settings for environment variables used by the trap forwarder.

2. The STDENV DD statement should specify a PDS that has a variable record format (RECFM=V or VB).

Configure the environment variable files for the Trap Forwarder

Example 4-11 shows the environment variables file for the Trap Forwarder.

Example 4-11 STDENV file for the TRAPFWDB task on each system

```
BROWSE    TCPIP.SC30.STDENV(TRPENV30) - 01.02          Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA                  1
RESOLVER_CONFIG=//'TCPIPA.TCPPARMS(DATAA30)'      2
TRAPFWD_CONF=//'TCPIPA.TCPPARMS(TRPCFG30)'        3

BROWSE    TCPIP.SC31.STDENV(TRPENV31) - 01.02          Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
TRAPFWD_CONF=//'TCPIPB.TCPPARMS(TRPCFG31)'
```

The following items explain Example 4-11:

1. _BPXK_SETIBMOPT_TRANSPORT establishes stack affinity to the indicated stack.

2. RESOLVER_CONFIG points to the TCPDATA resolver file for the same stack.

3. SNMPTRAP_DEST points to the optional trap destination file if needed by the agent

Configure the configuration files for the Trap Forwarder

Example 4-12 shows the forwarder's configuration file. The IP addresses in each file are the addresses of the *other* systems to which traps can be forwarded.

Example 4-12 TRAPFWD.CONF files for the TRAPFWDB task on each system

```
BROWSE    TCPIPA.TCPPARMS(TRPCFG30) - 01.00          Line 00000000 Col 001 080
 10.1.1.20  161
 10.1.1.30  161

BROWSE    TCPIPB.TCPPARMS(TRPCFG31) - 01.01           Line 00000000 Col 001 080
 10.1.1.10  161
 10.1.1.30  161
```

For Trap Forwarder setup and configuration details, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

4.2.3 Activation and verification of the z/OS SNMP agents

If the SNMP agent encounters any errors processing its configuration files, error messages are written to syslogd, not to the console. To activate the SNMP agent, the query engine, and the Trap Forwarder, enter a start command for each, as shown in Example 4-13.

Example 4-13 Starting the SNMP agent, the query engine, and the Trap Forwarder

```
S SNMPDB
$HASP373 SNMPDB   STARTED
SNMPDB issues:
   EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE

S SNMPQEB
$HASP100 SNMPQEB  ON STCINRDR
IEF695I START SNMPQEB  WITH JOBNAME SNMPQEB  IS ASSIGNED TO USER TCPIP   , GROUP  TCPGRP
$HASP373 SNMPQEB  STARTED
EZA6275I SNMP Query Engine running and awaiting queries...

S TRAPFWDB
TRAPFWD issues its own startup message, not related to SNMPD startup:
EZZ8409I TRAPFWD: INITIALIZATION COMPLETE
```

After verifying that each started task initializes successfully and issues the expected startup messages, use the NETSTAT CONN command to verify that each one is listening on its respective port, as shown in Example 4-14.

Example 4-14 Netstat CONN display of SNMPD, SNMPQE, and TRAPFWD

```
D TCPIP,TCPIPB,N,CONN
. . .
SNMPDB   0000280E LISTEN
  LOCAL SOCKET:   ::..1029
  FOREIGN SOCKET: ::..0
SNMPDB   0000280D UDP
  LOCAL SOCKET:   ::..161
  FOREIGN SOCKET: *..*
. . .
SNMPQEB  00002815 LISTEN
  LOCAL SOCKET:   0.0.0.0..1030
  FOREIGN SOCKET: 0.0.0.0..0
SNMPQEB  00002813 UDP
  LOCAL SOCKET:   0.0.0.0..6426
  FOREIGN SOCKET: *..*
SNMPQEB  00002814 UDP
  LOCAL SOCKET:   0.0.0.0..162
  FOREIGN SOCKET: *..*
. . .
TRAPFWDB 00002818 UDP
  LOCAL SOCKET:   ::..1162
  FOREIGN SOCKET: *..*
TRAPFWDB 00002819 UDP
  LOCAL SOCKET:   ::..6428
  FOREIGN SOCKET: *..*
. . .
```

4.3 z/OS SNMP subagents

A subagent extends the set of MIB variables supported by an SNMP agent. Each subagent must be set up and configured independently. The z/OS SNMP subagent and its relationship with the agent and the command client is illustrated in Figure 4-4 on page 210.

The z/OS SNMP subagent topics discussed in this section are:

- Description of SNMP subagents
- Configuration of SNMP subagents
- Activation and Verification of SNMP subagents

4.3.1 Description of SNMP subagents

A subagent extends the set of MIB variables supported by an SNMP agent. Each subagent must be set up and configured independently. The most common subagents associated with the z/OS Communications Server are discussed here.

TCP/IP subagent

The TCP/IP subagent in the z/OS Communications Server is a z/OS UNIX application that runs as a subtask in the TCP/IP address space.

Besides providing support for retrieval of TCP/IP stack management data, the TCP/IP subagent provides SET support, enabling remote configuration of some TCP/IP address space parameters. The TCP/IP subagent is configured and controlled by the SACONFIG statement in the PROFILE.TCPIP data set. If no SACONFIG statement is provided, then the TCP/IP subagent will be started at stack initialization. If you will not be retrieving TCP/IP stack management data, or you do not require the capability to do remote configuration of TCP/IP address space parameters, you can disable the subagent and save system resources.

OMPROUTE subagent

The OMPROUTE subagent implements the Open Shortest Path First (OSPF) MIB variable containing OSPF protocol and state information. The OMPROUTE subagent supports selected MIB objects defined in RFC 1850. The ROUTESA_CONFIG statement is used to enable the subagent support for OMPROUTE.

TN3270 Telnet subagent

The SNMP TN3270 Telnet subagent provides Telnet transaction data for monitored Telnet connections using the SNMP protocol. The TNSACONFIG statement in the TN3270 profile is used to enable the subagent support for TN3270 connections. The TNSACONFIG statement can only be coded within the TELNETGLOBALS block. The TN3270 SNMP MIB module (mvstn3270.mi2) defines performance data for TN3270 connections. The performance data is gathered only if the TN3270E Telnet server has been configured to monitor connections. To create and retrieve this data you must do the following:

- Define the MONITORGROUP and MONITORMAP profile statements to the TN3270E Telnet server, mapping the monitor parameters to client ID groups. See Example 4-27 on page 230 and Example 4-28 on page 231.
- Establish TN3270 connections where the client connections match the Client_Identifier parameter specified on one or more of the MONITORMAP profile statements.

Note: The TN3270 MIB objects are available only when monitoring has been defined within the TN3270 profile. If no monitoring has been defined in the profile, then the **osnmp**/**snmp** command will receive no information from its inquiry.

Network SLAPM2 subagent

The Network SLAPM2 (nslapm2) subagent contains the information that can be used to analyze network performance for respective policy. This subagent runs as its own started task and has SAF security requirements. See the security discussion earlier in this book in “Update RACF to define the SNMP agent started task” on page 212. For details about configuring this subagent, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

OSA-Express Direct subagent

The OSA product also provides an SNMP subagent that supports management data for OSA-Express features, called the *OSA-Express Direct subagent*. It can be used with the z/OS Communications Server SNMP support to retrieve management data.

An SNMP subagent exists on an OSA-Express feature, which is part of a direct path between the z/OS master agent (TCP/IP stacks) and an OSA-Express Management Information Base (MIB). The OSA-Express features support an SNMP agent by providing data for use by an SNMP management application, such as Tivoli NetView. This data is organized into MIB tables defined in the TCP/IP enterprise-specific MIB, and standard RFCs. The data is supported by the SNMP TCP/IP subagent. This subagent runs as its own started task and has SAF security requirements. See the security discussion in “Update RACF to define the SNMP agent started task” on page 212.

Additionally, IBM provides a support MIB for the OSA-Express Direct subagent outside of the z/OS Communications Server product and it must be acquired separately. The MIB data supported by the OSA-Express Direct subagent is defined in the OSA enterprise-specific MIB module, IBM-OSA-MIB. This MIB contains the SNMPv2 syntax for the OSA Direct specific MIB objects. The MIB module for the OSA-Express Direct subagent must match your System z server’s micro-code level (MCL). You can download the file from the IBM support website.

4.3.2 Configuration of SNMP subagents

Example 4-15 shows how to enable three of the subagents. For complete syntax and additional parameters for each of the statements used, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Example 4-15 Enabling subagents within their respective configuration profiles

```
TCP/IP stack subagent:
   SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161

OMPROUTE subagent:
   ROUTESA_CONFIG ENABLED=YES COMMUNITY="j0s9m2ap" AGENT=161;

TN3270E Telnet server subagent:
   TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
```

An example of the NSLAPM2 procedure JCL is shown in Example 4-16. We called our started task SLAPM2B.

Example 4-16 NSLAPM2 Proc JCL, SLAPM2B, subagent setup and its STDENV file

```
BROWSE    SYS1.PROCLIB(SLAPM2B) - 01.02              Line 00000000 Col 001 080
//SLAPM2B PROC STDENV=SLAENV&SYSCLONE.                      1
//SLAPM2B EXEC PGM=NSLAPM2,REGION=0M,TIME=NOLIMIT,
//       PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_CEE_ENVFILE=DD:STDENV")',
//            '/-o -c j02s9m2ap -P 161 -p TCPIPB')         2
//STDENV   DD DSN=TCPIP.SC31.STDENV(&STDENV.),DISP=SHR     3
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)

BROWSE    TCPIP.SC31.STDENV(SLAENV31) - 01.01              1
******************************** Top of Data ***
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
LIBPATH="/usr/lpp/tcpip/lib"
******************************** Bottom of Data *
```

The following items explain Example 4-16:

1. The STDENV member contains environment variable settings for NSLAPM2

2. The parameters tell NSLAPM2 what agent, community, port, and stack to use

3. The STDENV data set must have RECFM=V or VB.

The OSA-Express Direct subagent SNMP support for z/OS is provided by procedure IOBSNMP. This procedure is included as part of the z/OS Communications Server product. However, the MIB is not. You can update the catalogued procedure by copying the sample found in *hlq*.SEZAINST(IOBSNMP) to your system PROCLIB. Change the data set names as required to suit your local configuration. IOBSNMP has four optional parameters, as shown in Example 4-17. We called our started task SNMPOSAB.

Example 4-17 IOBSNMP Proc JCL, SNMPOSAB, setup information about the PARM statement

```
BROWSE    SYS1.PROCLIB(SNMPOSAB) - 01.00             Line 00000000 Col 001 080
//SNMPOSAB PROC PARMS='-d 0'
//IOBSNMP  EXEC PGM=IOBSNMP,TIME=1440,REGION=0M,DYNAMNBR=5,
//         PARM='-c j0s9m2ap -p 161 -s TCPIPB &PARMS.'
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
```

4.3.3 Activation and Verification of SNMP subagents

We started SLAPM2B using the MVS Start command:

```
S SLAPM2B
```

Typical startup messages for NSLAPM2 are shown in Example 4-18 on page 223.

Example 4-18 SLAPM2B startup messages

```
. . .
23.57.16 STC08401  IEF695I START SLAPM2B  WITH JOBNAME SLAPM2B  IS ASSIGNED TO U
23.57.16 STC08401  $HASP373 SLAPM2B  STARTED
23.57.16 STC08401  +EZZ8230I NSLAPM2 STARTING ON TCPIPB
23.57.16 STC08401  +EZZ8231I NSLAPM2 CONNECTED TO POLICY AGENT ON TCPIPB
....
main:  EZZ8230I NSLAPM2 STARTING ON TCPIPB
doPAPIConnect:  EZZ8231I NSLAPM2 CONNECTED TO POLICY AGENT ON TCPIPB
```

Several verification actions you can perform for the agent and forwarder are discussed in the following sections:

- Review the agent and subagent initialization messages
- Review IOBSNMP initialization and termination messages
- Review SNMPQE initialization and termination messages
- Interface display and traffic monitoring
- TCP/IP stack display and management
- SNMP agent configuration display and management
- Performance statistics displays and monitoring

Review the agent and subagent initialization messages

When the SNMP agent starts it issues an initialization complete message. The subagents also issue their own initialization complete messages. When the subagents successfully connect to the agent each one issues an additional message indicating connection to the agent. The messages indicate whether this is the first time they have connected to the agent while they have been executing or if this is a reconnect. A reconnect occurs if the SNMP agent terminates and then later reinitializes.

Example 4-19 shows samples of the initialization messages.

Example 4-19 SNMP agent and subagent initialization messages

```
S SNMPDB
$HASP373 SNMPDB   STARTED

SNMPDB issues:
   EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE

OMPB issues:
   EZZ8101I OMPROUTE SUBAGENT INITIALIZATION COMPLETE
   or
   EZZ8108I OMPROUTE SUBAGENT: RECONNECTED TO SNMP AGENT

TN3270B issues:
   EZZ6041I TELNET SNMP SUBAGENT INITIALIZATION COMPLETE
   or
   EZZ6043I TELNET SNMP SUBAGENT RECONNECTED TO SNMP AGENT

TCPIPB issues:
   EZZ3202I SNMP SUBAGENT: INITIALIZATION COMPLETE
   EZZ3221I SNMP SUBAGENT: SET REQUESTS DISABLED
   or
   EZZ3217I SNMP SUBAGENT: RECONNECTED TO SNMP AGENT

TRAPFWD issues its own startup message, not related to SNMPD startup:
   EZZ8409I TRAPFWD: INITIALIZATION COMPLETE
```

Example 4-20 shows samples of the termination messages of the agent and subagent.

Example 4-20 SNMP agent and subagent termination messages

```
P SNMPDB

SNMPDB issues:
   EZZ6204I SIGTERM RECEIVED FOR SNMP DAEMON WHICH IS NOW SHUTTING DOWN
   $HASP395 SNMPDB   ENDED

OMPB issues:
   EZZ8107I OMPROUTE SUBAGENT: CONNECTION TO SNMP AGENT DROPPED

TCPIPB issues:
   EZZ3216I SNMP SUBAGENT: LOST CONNECTION TO SNMP AGENT

TN3270B issues:
   EZZ6042I TELNET SNMP SUBAGENT LOST CONNECTION TO SNMP AGENT

IOBSNMP (SNMPOSAB) issues:
   IOB033E 09/29/2007 16:45:15 SNMP RC -5. Disconnecting from Agent
   IOB002I 09/29/2007 16:45:33 Could not obtain handle from agent. Exiting
```

Review IOBSNMP initialization and termination messages

An example of starting and ending the IOBSNMP started task is shown in Example 4-21. Note that IOBSNMP does not continue to execute if the SNMP agent task is not available or is taken down. The other subagents simply report the loss of contact, and wait until the agent returns. The IOBSNMP task must be started again after the SNMP task is restarted.

Example 4-21 IOBSNMP initialization and termination messages

```
S SNMPOSAB
IOB000I 09/29/2007 16:40:07 Starting OSA SNMP subagent
. . .
IOB028I 09/29/2007 16:40:07 Using stack name TCPIPB
IOB021I 09/29/2007 16:40:07 OSA SNMP subagent initialization complete

P SNMPOSAB
IOB031E 09/29/2007 16:55:24 OSA SNMP subagent has ended
```

Review SNMPQE initialization and termination messages

Both the SNMPQE task and the TRAPFWD task listen on port 162 by default. If you plan to run both, then set up TRAPFWD to listen on a different port. We ran our TRAPFWDB task on port 1162. The startup and shutdown SNMPQE messages are shown in Example 4-22.

Example 4-22 SNMPQE initialization and termination messages

```
S SNMPQEB
$HASP100 SNMPQEB  ON STCINRDR
IEF695I START SNMPQEB  WITH JOBNAME SNMPQEB  IS ASSIGNED TO USER TCPIP   , GROUP
TCPGRP
$HASP373 SNMPQEB  STARTED
EZA6275I SNMP Query Engine running and awaiting queries...

P SNMPQEB
$HASP395 SNMPQEB  ENDED
```

Interface display and traffic monitoring

See 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 227 for information about how to use the **snmp** command. By monitoring the amount of data transmitted over a router's interfaces, an administrator can monitor how many bytes of data have been transmitted between IP subnetworks in a particular period of time. You can also monitor the traffic size over an interface of an IP host that is running as an application server. The following are examples of MIB objects that can be used for this monitoring:

- ifInOctets gives the total number of octets received on the interface, including framing characters.
- ifOutOctets gives the total number of octets transmitted out of the interface, including framing characters.
- sysUpTime provides a count in one hundredths of a second of how long the SNMP agent has been running. That is how long the device has been up and running in most cases. This value is useful to determine the time differences from the previous collection and whether previous counter information is valid. Therefore, this value should be retrieved with each collection.

TCP/IP stack display and management

See 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 227 for information about how to use the **snmp** command. By changing particular MIB values, a TCP/IP stack configuration can be altered without recycling. Several of the operations supported are as follows:

- Change the IP forwarding attributes.
- Change the logic of the equal-cost multipath activation.
- Drop an existing TCP connection established to a remote IP host.
- Alter the buffer size allocated for each TCP connection or UDP association.

SNMP agent configuration display and management

See 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 227 for information about how to use the **snmp** command. The SNMPv3 framework allows you to change the SNMP agent's configuration dynamically. Several example operations supported are as follows:

- Change the security keys for SNMP managers.
- Add new users or SNMP managers.
- Change the group definition.

Performance statistics displays and monitoring

See 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 227 for information about how to use the **snmp** command. The SLA subagent maintains the SLA Performance Monitor MIB-2 (SLAPM2) that gives various performance information for each policy installed in a TCP/IP stack. Several possible monitoring options are as follows:

- Monitor the throughput.
- Monitor the transmission delay.
- Monitor the amount of data that meet a policy specification.

4.4 z/OS SNMP client command

The z/OS NetView SNMP and snmp commands act as a manager requesting information from an agent. Two SNMP client applications are provided with z/OS Communications Server:

- SNMP command from the NetView environment
- **snmp** command in the z/OS shell

4.4.1 Description of the SNMP client commands

The SNMP command in the NetView environment requires the use of the NetView product. It supports SNMP version 1. The **snmp**/**osnmp** command in the z/OS shell supports SNMP versions 1, 2, and 3. Depending on your requirements, you might decide to configure either or both of these clients, or to use an SNMP client on another platform. A brief description of the required implementation steps is given here. For complete details, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

4.4.2 Configuration tasks for the SNMP client commands

We discuss the configuration tasks for both the NetView SNMP command and the z/OS UNIX **snmp** command in this section.

Implementation of the NetView SNMP command

The NetView environment requires the set up of a number of items in the NetView address space, such as the SNMPIUCV task, the SNMP command processor, and the SNMP message definition file.

Perform the following tasks to prepare the NetView SNMP command:

1. Set up the SNMP Query Engine started task
2. Set up the SNMPARMS control member for NetView
3. Set up the hlq.MIBDESC.DATA data set

Set up the SNMP Query Engine started task

Update the SNMPQE cataloged procedure by copying the sample in SEZAINST(SNMPPROC) to your system PROCLIB. Specify SNMP parameters and change the data set names as required to suit your local configuration. Example 4-8 on page 217 shows a sample of the SNMPQE started task JCL.

Set up the SNMPARMS control member for NetView

SNMPIUCV reads the SNMPARMS member in the 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. Example 4-9 on page 217 shows NetView SNMPARMS for use with our query engine.

Set up the hlq.MIBDESC.DATA data set

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 SEZAINST(MIBDESC) to copy into your data set. When creating your *hlq*.MIBDESC.DATA data set, make sure that *hlq* matches what you have coded as the DATASETPREFIX in the TCPDATA resolver file.

Implementation of the osnmp z/OS UNIX command

The **osnmp** command is used to send SNMP requests to SNMP agents on local or remote hosts. The requests can be SNMPv1, SNMPv2, or SNMPv3. For SNMPv2 and SNMPv3 requests, the OSNMP.CONF configuration file is required. The *winSNMPname* specified on an OSNMP.CONF statement can be used as the value of the *-h* parameter on the **osnmp** command.

Perform the following tasks to prepare the **snmp**/**osnmp** command:

1. Set up snmp configuration information.
2. Set up user MIB object information.

Set up snmp 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, either `/etc/osnmp.conf` or `/etc/snmpv2.conf`, regardless of location, are the same. Only the first file found is used. A sample of this file is installed as `/usr/lpp/tcpip/samples/snmpv2.conf`. This sample should be copied and modified for your installation. The file we used is shown in Example 4-23.

Example 4-23 /etc/snmpv2.conf file used by the osnmp/snmp z/OS UNIX command

```
#-------------------------------------------------------------
# Community-based security  (SNMPv1 and SNMPv2c)
#-------------------------------------------------------------
v1       127.0.0.1   snmpv1
v2c      127.0.0.1   snmpv2c
v2c_ipv6 ::1         snmpv2c
mvs1     10.67.113.79 snmpv2c
sc30b    10.1.1.10 snmpv2c
sc31b    10.1.1.20 snmpv2c
sc32b    10.1.1.30 snmpv2c
# mvs2   mvs2c       snmpv2c    nosvipa
# mvs3   mvs3:1061   snmpv2c
mvs4     12ab::2     snmpv2c
```

Set up 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 **snmp** command using the MIBS.DATA file. A sample of the MIBS.DATA file is installed as `/usr/lpp/tcpip/samples/mibs.data`. Copy this sample to `/etc/mibs.data`, and modify it for your installation.

4.4.3 Using the osnmp/snmp z/OS UNIX command

See *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781, for information about how to use the **osnmp**/**snmp** command. Use the getbulk option of the **snmp** command to retrieve multiple objects with one command. A few examples of using the **snmp** command follow.

Tip: If you are interested in a single MIB object only, then you should use the **get** option with the fully qualified MIB object, instead of the **getbulk** option as shown in our examples.

Example 4-24 on page 228 shows system information. Notice that the Agent and active Subagents are represented in the list by the *sysORDescr.x* entries

Example 4-24 snmp system information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v getbulk 1.3.6.1.2.1.1
myDescr.0 = SNMPv3 agent version 1.0 with DPI version 2.0
myObjectid.0 = 1.3.6.1.4.1.2.3.13
myUptime.0 = 34300
myContact.0 = I&O Mainframe Network Support
myName.0 = SC30 - z/OS Communications Server
myLocation.0 = Datacenter Mainframe Operations
myServices.0 = 76
sysORLastChange.0 = 6500
sysORID.1 = 1.3.6.1.4.1.2.11.7.1
sysORID.2 = 1.3.6.1.4.1.2.11.7.2
sysORID.3 = 1.3.6.1.4.1.2.11.7.6
sysORID.4 = 1.3.6.1.4.1.2.11.7.3
sysORID.5 = 1.3.6.1.4.1.2.11.26.1
sysORDescr.1 = z/OS SNMP Agent
sysORDescr.2 = z/OS TCP/IP SNMP Subagent
sysORDescr.3 = z/OS TN3270 SNMP Subagent
sysORDescr.4 = z/OS OSPF SNMP Subagent
sysORDescr.5 = OSA subagent

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v getbulk 1.3.6.1.2.1.1
myDescr.0 = SNMPv3 agent version 1.0 with DPI version 2.0
myObjectid.0 = 1.3.6.1.4.1.2.3.13
myUptime.0 = 15400
myContact.0 = I&O Mainframe Network Support
myName.0 = SC31 - z/OS Communications Server
myLocation.0 = Datacenter Mainframe Operations
myServices.0 = 76
sysORLastChange.0 = 700
sysORID.1 = 1.3.6.1.4.1.2.11.7.1
sysORID.2 = 1.3.6.1.4.1.2.11.26.1
sysORID.3 = 1.3.6.1.4.1.2.11.7.6
sysORID.4 = 1.3.6.1.4.1.2.11.7.2
sysORID.5 = 1.3.6.1.4.1.2.11.7.3
sysORDescr.1 = z/OS SNMP Agent
sysORDescr.2 = OSA subagent
sysORDescr.3 = z/OS TN3270 SNMP Subagent
sysORDescr.4 = z/OS TCP/IP SNMP Subagent
sysORDescr.5 = z/OS OSPF SNMP Subagent
CS07 @ SC30:/u/cs07>
```

We established a TN3270 client connection from the SC31 TSO session to the SC30 TN3270E Telnet server. From SC30's perspective the local socket would be 10.1.1.10 and the Foreign socket would be 10.1.1.20, as shown in Example 4-25.

Example 4-25 snmp stack information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v getbulk ibmMvsTcpipProcname
ibmMvsTcpipProcname.0 = TCPIPB
ibmMvsTcpipAsid.0 = 111

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v getbulk ibmMvsTcpipProcname
ibmMvsTcpipProcname.0 = TCPIPB
ibmMvsTcpipAsid.0 = 132
```

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 30 getbulk tcpConnectionEntry
tcpConnectionState.1.4.10.1.1.10.23.1.4.10.1.1.20.1029 = 5
tcpConnectionState.1.4.10.1.1.10.1056.1.4.10.1.1.10.1055 = 8
tcpConnectionState.1.4.10.1.1.10.1062.1.4.10.1.1.10.1061 = 8
tcpConnectionState.1.4.10.1.8.11.992.1.4.10.1.6.21.1035 = 5
tcpConnectionState.1.4.127.0.0.1.1024.1.4.127.0.0.1.1025 = 5
tcpConnectionState.1.4.127.0.0.1.1025.1.4.127.0.0.1.1024 = 5
tcpConnectionState.1.4.127.0.0.1.1081.1.4.127.0.0.1.1082 = 5
tcpConnectionState.1.4.127.0.0.1.1082.1.4.127.0.0.1.1081 = 5

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v -m 30 getbulk tcpConnectionEntry
tcpConnectionState.1.4.10.1.1.20.1029.1.4.10.1.1.10.23 = 5
tcpConnectionState.1.4.127.0.0.1.1024.1.4.127.0.0.1.1025 = 5
tcpConnectionState.1.4.127.0.0.1.1025.1.4.127.0.0.1.1024 = 5
tcpConnectionState.1.4.127.0.0.1.1026.1.4.127.0.0.1.1027 = 5
tcpConnectionState.1.4.127.0.0.1.1027.1.4.127.0.0.1.1026 = 5
```

Example 4-26 shows omproute/ospf information.

Example 4-26 snmp Omproute/OSPF information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 10 getbulk ospf
ospfRouterId.0 = 10.1.1.10
ospfAdminStat.0 = 1
ospfVersionNumber.0 = 2

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v -m 10 getbulk ospf
ospfRouterId.0 = 10.1.1.20
ospfAdminStat.0 = 1
ospfVersionNumber.0 = 2

CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 10 getbulk ospfIfTable
ospfIfIpAddress.10.1.1.10.0 = 10.1.1.10
ospfIfIpAddress.10.1.2.11.0 = 10.1.2.11
ospfIfIpAddress.10.1.2.12.0 = 10.1.2.12
ospfIfIpAddress.10.1.3.11.0 = 10.1.3.11
ospfIfIpAddress.10.1.3.12.0 = 10.1.3.12
ospfIfIpAddress.10.1.4.11.0 = 10.1.4.11
ospfIfIpAddress.10.1.5.11.0 = 10.1.5.11
ospfIfIpAddress.10.1.6.11.0 = 10.1.6.11
ospfIfIpAddress.10.1.8.11.0 = 10.1.8.11
ospfIfIpAddress.10.1.8.12.0 = 10.1.8.12

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v -m 10 getbulk ospfIfTable
ospfIfIpAddress.10.1.1.20.0 = 10.1.1.20
ospfIfIpAddress.10.1.2.21.0 = 10.1.2.21
ospfIfIpAddress.10.1.2.22.0 = 10.1.2.22
ospfIfIpAddress.10.1.3.21.0 = 10.1.3.21
ospfIfIpAddress.10.1.3.22.0 = 10.1.3.22
ospfIfIpAddress.10.1.4.21.0 = 10.1.4.21
ospfIfIpAddress.10.1.5.21.0 = 10.1.5.21
ospfIfIpAddress.10.1.6.21.0 = 10.1.6.21
ospfIfIpAddress.10.1.8.21.0 = 10.1.8.21
ospfIfIpAddress.10.1.8.22.0 = 10.1.8.22
```

Example 4-27 shows TN3270 information.

Example 4-27 snmp TN3270E Telnet server information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 40 getbulk ibmMvsTN3270ConnTable
ibmMvsTN3270ConnStartTime.1.4.10.1.1.10.23.1.4.10.1.1.20.1031=2007-09-29,0:0:58.8
ibmMvsTN3270ConnAppl.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = SC30TS02
ibmMvsTN3270ConnLuName.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = SC30BB01
ibmMvsTN3270ConnLogMode.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = SNX32704
ibmMvsTN3270ConnProto.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = '02'h
ibmMvsTN3270ConnRtGroupIndex.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 1
ibmMvsTN3270ConnRtIpMethod.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtAvgRt.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtAvgIpRt.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtAvgCountTrans.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 8
ibmMvsTN3270ConnRtIntTimeStamp.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 2007-09
-29,0:6:58.8
ibmMvsTN3270ConnRtTotalRts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtTotalIpRts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtCountTrans.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 12
ibmMvsTN3270ConnRtCountIP.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtElapsRndTrpSq.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtElapsIpRtSq.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtElapsSnaRtSq.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket1Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 12
ibmMvsTN3270ConnRtBucket2Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket3Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket4Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket5Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270MonGroupName.1 = SNAONLY
ibmMvsTN3270MonGroupName.2 = SNAANDIP
ibmMvsTN3270MonGroupType.1 = '30'h
ibmMvsTN3270MonGroupType.2 = 'f0'h
ibmMvsTN3270MonGroupSampPeriod.1 = 120
ibmMvsTN3270MonGroupSampPeriod.2 = 120
ibmMvsTN3270MonGroupSampMult.1 = 5
ibmMvsTN3270MonGroupSampMult.2 = 5
ibmMvsTN3270MonGroupBucketBndry1.1 = 100
ibmMvsTN3270MonGroupBucketBndry1.2 = 100
ibmMvsTN3270MonGroupBucketBndry2.1 = 200
ibmMvsTN3270MonGroupBucketBndry2.2 = 200
ibmMvsTN3270MonGroupBucketBndry3.1 = 300
ibmMvsTN3270MonGroupBucketBndry3.2 = 300
ibmMvsTN3270MonGroupBucketBndry4.1 = 400
ibmMvsTN3270MonGroupBucketBndry4.2 = 400
ibmProd.188.1.1.1.1.12 = '0002'h
CS07 @ SC30:/u/cs07>
```

The TN3270E Telnet server profile must have MONITORGROUP and MONITORMAP statements defining the performance statistics, which should be collected for the mapped connections.

Note: The TN3270 MIB objects are available only when monitoring has been defined within the TN3270 profile. If no monitoring has been defined in the profile, then the **osnmp**/**snmp** command will receive no information to its inquiry.

The statements in our TN3270 profile, TELNB30B, are shown in Example 4-28.

Example 4-28 TN3270E Telnet server profile statements defining the MONITORGROUP for port 23

```
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS
MONITORGROUP SNAONLY
    AVERAGE
    BUCKETS
    NODYNAMICDR
    NOINCLUDEIP
    AVGSAMPPERIOD 120
    AVGSAMPMULTIPLIER 5
    BOUNDARY1 100
    BOUNDARY2 200
    BOUNDARY3 300
    BOUNDARY4 400
  ENDMONITORGROUP

  DESTIPGROUP ALLUSERS    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  MONITORMAP SNAONLY    DESTIPGRP,ALLUSERS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL SC30N*          ; NetView
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
```

The profile statements defining the MONITORGROUP for port 992 are shown in Example 4-29.

Example 4-29 TN3270E Telnet server profile statements defining the MONITORGROUP for port 992

```
MONITORGROUP SNAANDIP
  AVERAGE
  BUCKETS
  DYNAMICDR
  INCLUDEIP
  AVGSAMPPERIOD 120
  AVGSAMPMULTIPLIER 5
  BOUNDARY1 100
  BOUNDARY2 200
  BOUNDARY3 300
  BOUNDARY4 400
ENDMONITORGROUP

DESTIPGROUP GENERALUSER 10.1.8.11   ENDDESTIPGROUP
DESTIPGROUP ADMIN       10.1.8.12   ENDDESTIPGROUP
DESTIPGROUP PAYROLL     10.1.8.23   ENDDESTIPGROUP
DESTIPGROUP SHIPPING    10.1.1.10   ENDDESTIPGROUP
```

```
DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

MONITORMAP SNAANDIP   DESTIPGRP,GENERALUSER
PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER
```

Additional SNMP MIB objects

In previous versions, TCP/IP only used SNMP counters for the number of the accepted TCP connections in the *ibmTcpipMvsTcplistenerTable* MIB table. There was one set of counters per server represented in the table defined in the IBM MVS TCP/IP enterprise-specific MIB modules shipped with z/OS Communications Server and supported by the TCP/IP subagent.

To determine the total count of accepted connections for a TCP/IP stack management applications had to retrieve the accepted connection counter for each server and sum these counts. Accepted connections counters for servers which had terminated were no longer available.

The z/OS Communications Server provides two specific counters in the IBM MVS TCP/IP enterprise-specific MIB module:

- `ibmMvsTcpAcceptCount` (32-bit counter)
- `ibmMvsTcpHCAcceptCount` (64-bit counter)

Both counters provide the total number of connections accepted by all listeners. They differ only on the size. Example 4-30 shows samples that we obtained from **snmp** commands in z/OS UNIX.

Example 4-30 Examples of TCP layer accepted connections counters

```
CS07@SC30:/u/cs07>snmp-h10.1.1.20-cj0s9m2ap-vgetbulkibmMvsTcpHCAcceptCount
ibmMvsUdpLastAct.0.0.0.0.161 = 0
CS07@SC30:/u/cs07>snmp-h10.1.1.20-cj0s9m2ap-vgetbulkibmMvsTcpAcceptCount
ibmMvsTcpAcceptCount.0 = 1
CS07 @ SC30:/u/cs07>cd \
```

Another object available in the IBM MVSTCP/IP enterprise-specific MIB module is ibmMvsCpcNd. The central processing complex node descriptor value for the central processing complex on which the TCP/IP subagent is active. You can use this value to uniquely identify this CPC. If the node descriptor cannot be obtained, then this MIB object is set to a zero-length string.

Example 4-31 shows that the object *ibmMvsCpcNd* can be obtained either through the **snmp** command or through the MVS console command D M=CPU.

Example 4-31 The ibmMvsCpcNd obtained in two different ways

```
CS05 @ SC30:/u/cs07>snmp -h 10.1.1.20 -c j0s9m2ap -v getbulk ibmMvsCpcNd
ibmMvsCpcNd.0 = '0f0f2f0f9f4e2f1f8c9c2d4f0f2f0f0f0f0f0f0f0f2f9f9f1c5fff0'h
CS07 @ SC30:/u/cs07>
. . . . . .
D M=CPU
IEE174I 10.46.20 DISPLAY M 364
PROCESSOR STATUS
ID  CPU                 SERIAL
00  +                    113BD52817
01  +                    113BD52817
02  +A                   113BD52817
```

```
03  +I                    113BD52817
04  -
05  -
06  -A
07  -I

CPC ND = 002817.M32.IBM.02.0000000B3BD5
CPC SI = 2817.715.IBM.02.00000000000B3BD5
         Model: M32
CPC ID = 00
CPC NAME = SCZP301
LP NAME = A11        LP ID = 11
CSS ID  = 1
MIF ID  = 1

+ ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
N NOT AVAILABLE

A        APPLICATION ASSIST PROCESSOR (zAAP)
I        INTEGRATED INFORMATION PROCESSOR (zIIP)
CPC ND  CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI  SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID  CENTRAL PROCESSING COMPLEX IDENTIFIER
CPC NAME CENTRAL PROCESSING COMPLEX NAME
LP NAME  LOGICAL PARTITION NAME
LP ID    LOGICAL PARTITION IDENTIFIER
CSS ID   CHANNEL SUBSYSTEM IDENTIFIER
MIF ID   MULTIPLE IMAGE FACILITY IMAGE IDENTIFIER
```

Removal of DPI API 4KB buffer restriction

Previously, there were restrictions when using Distributed Protocol Interface (DPI) API for any SNMP subagent that connected to the z/OS SNMP agent to provide management data. The agent used DPI API to send SNMP requests to subagents and the subagents used the DPI API to send the agent responses to SNMP requests and traps. The DPI API had a hard-coded limit of 4 KB buffer-size. Because of the buffer-size, the management applications using SNMP GETNEXT, GETBULK, or BULKWALK often receive a “too big” error on requests and could not manage the information for the clients.

To allow larger amounts of data to be exchanged between SNMP agent and subagent, DPI API now supports up to 65535 byte buffers and the agents support up to 65535 bytes for responses.

Important: For subagents that are not provided by z/OS Communications Server, the subagent load modules must be linked again to include the latest DPI API functions.

As a result, the applications can retrieve more data per SNMP request, especially when using the SNMP GETBULK and BULKWALK options.

Options for message EZZ6317I

SNMP message EZZ6317I was created to warn that support for configuration of the SNMP sysObjectId value would be discontinued in a future release; however, the message was only written to the syslog and it was not always noticed. The sysObjectId MIB object should not be

set to other than its default value because it identifies the z/OS Communications Server agent to network management applications.

An option exists to enable the message EZZ6317I to be written to the console and syslog daemon, making the it more visible to those customers currently configuring this MIB object.

Sample of OSNMPD.DATA information is shipped with the z/OS Communications Server product and installed as file `/usr/lpp/tcpip/samples/osnmpd.data`. OSNMPD.DATA information can be used to configure values of some of the MIB objects supported by the SNMP agent.

Example 4-32 illustrates the sample file where the object sysObjectId was commented out and, in this way, you do not have the message written to the console nor to syslogd.

Example 4-32 OSNMPD.DATA without sysObjectId

```
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5694-A01
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "Unknown"
  sysLocation "Unknown"
  sysName "z/OS Communications Server"
# Default value of sysObjectID is equivalent to ibmTcpIpMvs
#  in the ibmAgents subtree; this is the sysObjectID representing
#  IBM z/OS Communications Server
# Changing this value is not recommended, as it is intended to allow
#  network management applications to identify this agent as the
#  z/OS Communications Server SNMP agent.  The ability to change it
#  will be disabled in a subsequent release.
# sysObjectID     "1.3.6.1.4.1.2.3.13"
  snmpEnableAuthenTraps 1
  saDefaultTimeout    6
  saMaxTimeout  700
# saAllowDuplicateIDs must be set to 1 to allow multiple DPI version 1
#  subagents
  saAllowDuplicateIDs   1
  dpiPathNameForUnixStream "/tmp/dpi_socket"
# Default value of sysServices indicates support for
#  internet, end-to-end, and application layers as
#  defined in RFC 1907.
  sysServices       76
```

Example 4-33 is before changing the OSNMPD.DATA file. We did not receive the EZZ6317I message.

Example 4-33 Before changing the OSNMPD.DATA file

```
S SNMPD
$HASP100 SNMPD    ON STCINRDR
IEF695I START SNMPD    WITH JOBNAME SNMPD    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 SNMPD    STARTED
EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE
EZZ3217I SNMP SUBAGENT: RECONNECTED TO SNMP AGENT
```

Example 4-34 shows the startup messages after changing the OSNMPD.DATA file, which enabled the sysObjectId.

Example 4-34 After changing the OSNMPD.DATA file

```
S SNMPD
$HASP100 SNMPD    ON STCINRDR
IEF695I START SNMPD    WITH JOBNAME SNMPD    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 SNMPD    STARTED
EZZ6317I CONFIGURATION OF SYSOBJECTID ACCEPTED BUT WILL NOT BE
ALLOWED IN FUTURE RELEASES
EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE
```

4.5 Problem determination for the SNMP facilities

The z/OS UNIX **osnmp** command provides the SNMP manager function from the z/OS shell to query SNMP agents for network management information. Use the **osnmp** command to issue SNMP requests to agents and to process SNMP responses returned by agents. The z/OS UNIX **osnmp** command supports SNMPv1, SNMPv2c, and SNMPv3 requests.

snmp is a synonym for the **osnmp** command in the z/OS UNIX shell. **snmp** command syntax is the same as that for the **osnmp** command.

For a comprehensive discussion on the use of the **osnmp** command and its syntax, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Certain MIB values are useful for problem determination in IP networks to solve problems such as performance problems or lack of connectivity.

Examples of MIB objects that can be used to determine problems in the data-link interface (if) layer are as follows:

- ifInErrors, ifOutErrors: give the number of inbound/outbound frames that were discarded because of errors.
- ifInDiscards, ifOutDiscards: give the number of inbound/outbound frames that were discarded because of resource limitations.

Examples of MIB objects that can be used to determine problems in the IP layer are as follows:

- ipInHdrErrors: Gives the number of IP packets that were discarded because of errors in their IP headers.
- ipInUnknownProtos: Indicates the number of IP packets that were received successfully but discarded because of either unknown or unsupported protocol.
- ipInDiscards, ipOutDiscards: Give the number of inbound/outbound IP packets that were discarded because of resource limitations.
- ipReasmFails: Provides the number of failures detected by the IP re-assembly algorithm (for whatever reason: timed out, errors, and so forth).

Certain MIB objects are useful to determine problems related to the TCP or UDP protocol:

- tcpRetransSegs: Gives the number of TCP segments retransmitted.
- udpInErrors: Gives the number of UDP datagrams discarded for reasons other than the lack of an application at the destination port.

4.6 Additional information sources for SNMP

See the following sources for additional information about SNMP protocols:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *OSA-Express Implementation Guide*, SG24-5948
- *TCP/IP Tutorial and Technical Overview*, GG24-3376

Tip: For descriptions of security considerations that affect specific servers or components, see "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

5

IP printing

IP printing supports the sending of print output from a client device on an IP network to a printer or a print server gateway on that network. This chapter focuses on the IP printing functions that are available in the z/OS Communications Server.

The protocols defined in Request for Comment (RFC) 1179 and its amendments form the basis for printing in a TCP/IP network. The line printer daemon (LPD) is the remote print server and the line printer requester (LPR) is the client defined by this protocol. If the destination printer is a network printer, one defined to the LPD with an IP address or a DNS name, then the LPD uses the TCP protocol to forward the file to the final IP destination.

This chapter discusses the following IP printing topics.

Section	Topic
5.1, “Conceptual overview of IP printing” on page 238	The basic concepts of IP printing.
5.2, “LPR/LPD” on page 240	Commonly implemented LPR/LPD scenarios, their dependencies, advantages, and configuration examples.
5.3, “Infoprint Server” on page 245	An overview of Infoprint Server and its components. Sample setup ISPF panels are listed (implementation or verification procedures are not shown because they are beyond the scope of this publication).
5.4, “Problem determination for LPR/LPD” on page 255	Problem determination procedures for the LPR/LPD scenario.
5.5, “Additional information sources for IP printing” on page 261	References to additional documentation about IP printing.

5.1 Conceptual overview of IP printing

As illustrated in Figure 5-1, the line printer requester (LPR) and the line printer daemon (LPD) are standard applications provided with the z/OS Communications Server that support IP Printing Services. The Line Print Requester (client) and the Line Print Daemon (LPD) use the Pascal application programming interface to send output data directly to the TCP/IP protocol layers, bypassing the socket interfaces of the Logical and Physical file systems of z/OS UNIX.

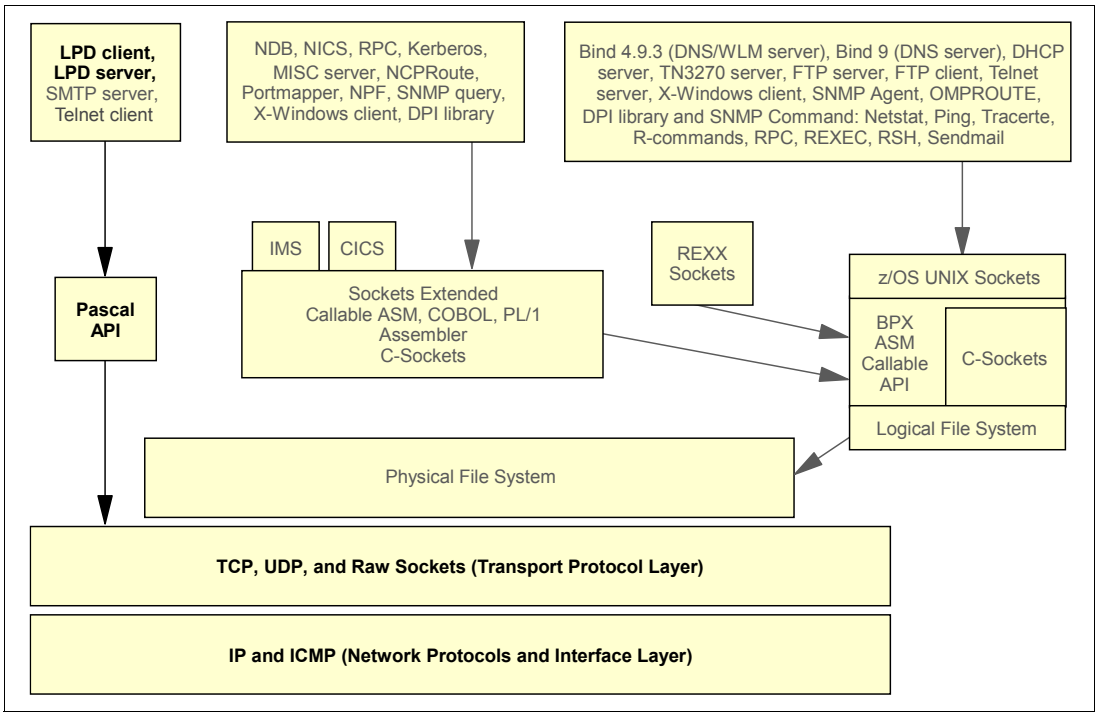


Figure 5-1 z/OS IP Printing Services

Many printers can support LPD themselves, allowing network clients to print directly to them. Also, with advanced implementations, print services can be provided by a dedicated print management solution that provides robust management and administration capabilities for many printers.

This section discusses the following concepts:

- What is IP printing
- How does IP Printing work
- How can IP Printing be applied

5.1.1 What is IP printing

The basic capabilities of LPR are limited. LPR is a good testing tool for verifying communications to a print server and for verifying print delivery to a printer controlled by that server. However, if advanced formatting features or print file management is required, additional products are needed. Printing products such as the IBM Infoprint Server extend the options for the management of printing in your organization. The Infoprint Server is the IBM strategic z/OS IP print management product that utilizes IP protocols to deliver centralized enterprise-wide print management. By combining the flexible Infoprint Server interfaces with the robust spool management capabilities of the Job Entry Subsystem (JES) you can achieve state-of-the-art print operations for most IP printing needs.

No future functional enhancements are planned for the IBM Network Print Facility (NPF) for z/OS, the predecessor of Infoprint Server.

5.1.2 How does IP Printing work

The diagram in Figure 5-2 shows the relationship of LPR to LPD.

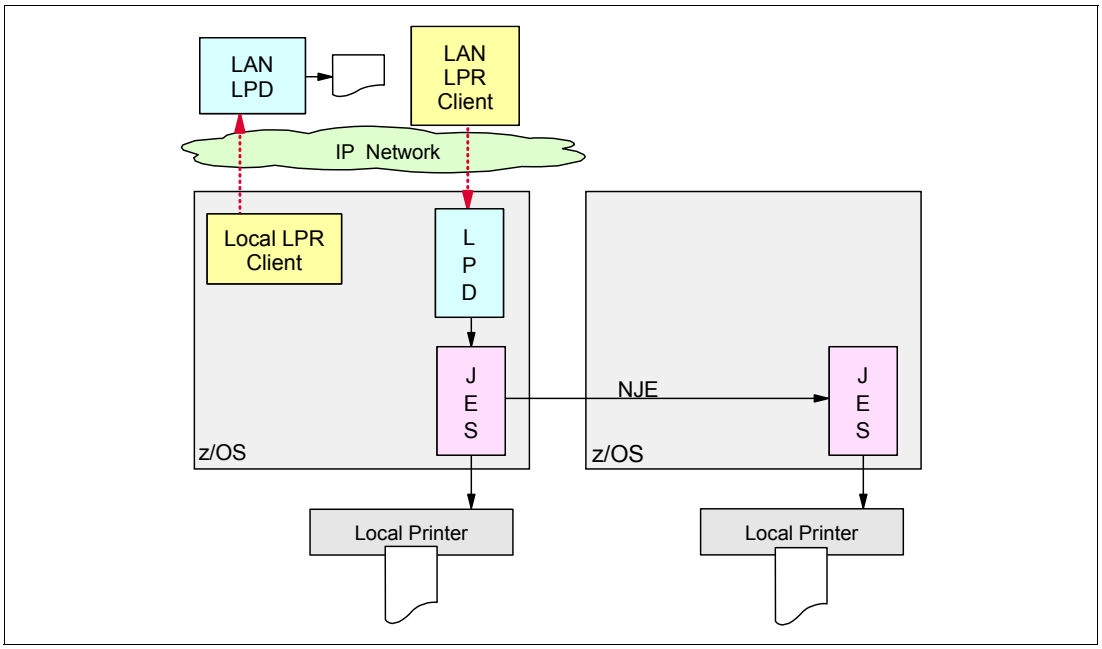


Figure 5-2 LPR/LPD relationships

In a simple LPR/LPD environment with no additional print management products, a client can issue the LPR command to print a file. By specifying a few parameters on the LPR command, the client can direct the output to a printer defined on a print server that is acting as the LPD server. Optional LPR parameters can be used to manipulate the printed file format and to instruct the LPD server to perform additional processing after the file arrives at the server. Two files are transmitted from the LPR client to the LPD server for each file to be printed:

- A control file contains structured parameter settings such as number of copies, special forms, special fonts, target printer, queue class, and many others.
- The actual data file to be printed.

The LPD server is responsible for managing the print queues for printers it has defined. It is also responsible for the integrity of the received files and for successfully printing them. When it is accepting print for a mainframe-attached printer (local system printer or remote printer), the z/OS LPD server uses the JES spool as its repository and takes advantage of spool integrity management provided by JES.

5.1.3 How can IP Printing be applied

IP printing can be accomplished by using a simple LPR/LPD implementation. More advanced printing packages can be used to provide advanced capabilities, such as documenting and managing printer inventory, providing data stream transforms, and providing a web-based application for help desk operators to access print queue status and printer information.

With the advent of TCP/IP networking and newer information technologies, printing requirements are changing. For example:

- Applications and workstation users need the flexibility to print to any printer (network printers and host printers).

  Businesses that print statements (such as banking statements, invoices, and bills of materials) need to print both on network-attached printers and on higher-volume, host-attached printers.

- Users need easy-to-use software.

  Users want graphical interfaces (GUIs) to handle the complex tasks of printing and managing printers.

- Companies require more print server capacity.

  Companies with a combination of stand-alone and host-connected printers need more print server capacity to meet their distributed printing needs.

These requirements introduce new issues:

- How to handle the wide range of printers and formatting options available in an environment and enable users of traditional terminals and distributed workstations share those printers.
- How to support print from applications that are consolidated into a z/OS environment without re-engineering their printing functions.
- How to reduce costs by consolidating print servers.

Possible solutions for these issues can be achieved by using the simple LPR/LPD functions, or by implementing the more sophisticated Infoprint Server facilities.

When a simple LPD server is required for inbound print

If your printing requirements are to provide an LPD server on the z/OS platform with little or no use of the outbound LPR function, then a simple implementation of the LPD server will achieve a quick solution. This supports inbound print traffic that is to be printed by the mainframe. However, it will be limited in functionality.

When manageability of centralized printing is required

If outbound print traffic is a requirement, then the management of that printing cannot be provided with the simple LPR function. A more robust solution will be necessary to provide queue management and reporting. In addition, you might be faced with a requirement to reformat print files before they are delivered to printers requiring special data streams. A printing solution such as the IBM Infoprint Server should be considered.

5.2 LPR/LPD

A simple Line Print Daemon (LPD) server can be implemented on the mainframe to enable the z/OS platform as a print server. This section discusses the following topics:

- Description of LPR/LPD
- Configuration tasks for LPR/LPD
- Activation and verification of LPR/LPD

5.2.1 Description of LPR/LPD

You can use LPD with the TSO LPR command or batch LPR jobs to support TCP/IP print. This type of setup is limited in functionality, but involves minimal effort and planning.

A TSO session user can enter the client LPR command manually for each file to be printed. A batch job can also be set up to execute the LPR command in a background batch TSO environment. The job can be scheduled and automated through a job scheduling system, but there is no centralized management of the print files.

Dependencies of LPR/LPD

The z/OS LPD server uses JES to manage print files. The LPD server can receive the print files being forwarded to it by the various LPR clients in the network. If the destination printer is a z/OS controlled printer, the print file is placed onto the JES spool. Then JES assumes responsibility for sending it to the printer.

The 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.

Considerations for using LPR/LPD

The LPR command parameters provide only limited functionality. There is no queue management interface with the LPR function and it has a limited query capability. The LPD server function also does not provide a robust queue management interface. Because it immediately places print files that are destined to mainframe controlled printers onto the JES spool, the only user query interface it provides is the limited LPQ client command, which enables the client to query the status of print jobs. However, it seldom has anything to show the client. After the file is under JES control, the LPD considers the file as having been delivered.

5.2.2 Configuration tasks for LPR/LPD

Use Table 5-1 to determine what print function you need to customize based on the origin of the print request. TSO interactive users and TSO batch jobs can use the LPR client function (LPR command) to print on IP network printers. Set up the LPD server on your z/OS system by creating an LPD configuration data set to enable remote LPR clients to use printers on your local JES system and Network Job Entry (NJE) network.

Table 5-1 LPR and LPD function table

Origin	Destination	Function
TSO user or batch Job	TCP/IP network printer	LPR
TCP/IP network client	z/OS JES or NJE printer	LPD

The line printer daemon (LPD) is the printer server that enables other hosts in your TCP/IP network to use printers on your z/OS system. You start the LPD server as an address space in your local system. The LPD server enables users in your TCP/IP network to address JES-controlled printers. A client from any TCP/IP host can use the local line printer requester (LPR) command to print a local file on a JES-controlled printer. The printer can be a local JES system printer, or it can be a printer accessed through an NJE network.

The LPR command enables a user on the local host to submit a file for printing on a remote printer server. Support is included for proper translation of carriage control information if the file you want to print uses formatted carriage control characters in the file's records. If you

have a PostScript file on the z/OS system, you can use the LPR command to print that file on a PostScript printer in the TCP/IP network. *z/OS Communications Server: IP Configuration Guide*, SC31-8775, has a comprehensive chapter on setting up the LPD print server. The checklist in that chapter should be used to accomplish successful setup.

The configuration tasks for the LPD server are:

- Update TCP/IP Profile data set with AUTOLOG and PORT statements
- Customize the LPSERVE proc from SEZAINST(LPSPROC)
- Update LPD server configuration file from SEZAINST(LPDDATA)

Update TCP/IP Profile data set with AUTOLOG and PORT statements

An example of the AUTOLOG and PORT statements in the PROFILE data set is shown in Example 5-1.

Example 5-1 AUTOLOG and PORT statements for LPSERVE

```
AUTOLOG
   LPSERVEB
ENDAUTOLOG

PORT 515 TCP LPSERVEB
```

Update the LPD server cataloged procedure shipped as SYS1.SEZAINST(LPSPROC), but with an actual procedure name of LPSERVE.

Customize the LPSERVE proc from SEZAINST(LPSPROC)

An example of the cataloged procedure for LPSERVE is shown in Example 5-2.

Example 5-2 Cataloged procedure for LPSERVE

```
//LPSERVEB PROC MODULE=LPD,
//         LPDDATA=TCPIPB.TCPPARMS(LPDATB&SYSCLONE.),
//         LPDPRFX='PREFIX TCPIP',
//         DIAG=''
//*
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//LPD   EXEC PGM=MVPMAIN,
//  PARM=('&MODULE,ERRFILE(SYSERR),HEAP(512)',
//       'NOSPIE/ ''&LPDDATA'' &LPDPRFX &DIAG'),
//       REGION=6M,TIME=1440
//SPOOL OUTPUT CHARS=GT12
//STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR
//LPD1 OUTPUT CHARS=GT12
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.)
```

Update LPD server configuration file from SEZAINST(LPDDATA)

An example of the configuration data set, LPDDATA, for LPSERVE is shown in Example 5-3.

Example 5-3 The LPDDATA configuration data set for LPSERVE

```
BROWSE    TCPIPB.TCPPARMS(LPDATB31) - 01.00          Line 00000000 Col 001 080
;LPD CONFIGURATION DATA SET
;==========================
;
;DEBUG
;
 SERVICE sysprt1 PRINTER
    LOCAL
    FILTERS f l p r
    LINESIZE 132
    PAGESIZE 60
;
 SERVICE njeprt1 PRINTER
  NJE DEST=BRANCH22 IDENTIFIER=RMT2 OUTPUT=LPD1
  FILTERS f l p r
  LINESIZE 132
  PAGESIZE 60
;
 SERVICE lanprt4 PRINTER
  REMOTE pokip1145@lanprt.itso.ibm.com
; FAILEDJOB MAIL
;
SERVICE pun1 PUNCH
   LOCAL
   FILTERS l
   LINESIZE 80
;
;
OBEY CSO2 CSO1
```

5.2.3 Activation and verification of LPR/LPD

Example 5-4 shows the startup messages for LPSERVE.

Example 5-4 LPSERVEB startup messages

```
S LPSERVEB
$HASP100 LPSERVEB ON STCINRDR
IEF695I START LPSERVEB WITH JOBNAME LPSERVEB IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 LPSERVEB STARTED
EZY1876I LPD STACK FUNCTIONS STARTED WITH PARAMETER LPD,ERRFILE(SYSERR
),HEAP(512),NOSPIE/ 'TCPIPB.TCPPARMS(LPDATB31)' PREFIX TCPIP .
```

A display of the TCP/IP stack connections shows the LPSERVE listener in Example 5-5.

Example 5-5 LPSERVEB listening on port 515

```
/d tcpip,tcpipb,n,conn
RESPONSE=SC31
 EZD0101I NETSTAT CS V1R13 TCPIPB 688
 USER ID  CONN     STATE
 FTPDB1   0000002F LISTEN
   LOCAL SOCKET:   ::..21
   FOREIGN SOCKET: ::..0
 JES2S001 00000013 LISTEN
   LOCAL SOCKET:   ::..175
   FOREIGN SOCKET: ::..0
 LPSERVEB 000005C8 LISTEN
   LOCAL SOCKET:   0.0.0.0..515
   FOREIGN SOCKET: 0.0.0.0..0
```

In the following command discussions, the *printer* name is a printer defined on the *host* at the indicated location. The location can be either the IP address or the DNS name of the print server for the printer, and can be any print server in the network: either a z/OS system where LPSERVE is running as a started task, or a non-z/OS print server platform in the network.

The TSO LPR-related commands are:

- LPQ: Used to query print job and queue status on a remote LPD server
- LPRM: Used to remove one or more print jobs from the queue on a remote LPD server
- LPR: Used to send a file to a remote LPD server to be printed
- LPRSET: Used to set a default destination printer and print server, or to show the default

The LPQ command can query the status for all jobs on the printer's queue, ask for detailed status information related to those jobs, or query the status of jobs on the printer queue for a specific user ID or job number, as shown in Example 5-6.

Example 5-6 LPQ samples

```
LPQ (ALL PRINTER sysprt1 AT 10.1.1.20
LPQ (PRINTER lanprt4 AT wtsc31b.itso.ibm.com TRACE
LPQ myuserid (PRINTER nejprt1 AT 10.1.1.20
LPQ 273 (PRINTER lanprt4 AT wtsc31b.itso.ibm.com
```

See the *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, for a complete list of parameters that can be specified on the LPR command. Example 5-7 shows samples of the LPR command.

Example 5-7 LPR samples

```
LPR 'TESTFILE.PRINT' (PRINTER testprt1 AT lpdsrvr1 CLASS v COPIES 2 FILTER f l p r
                JOB DEST=njenode3,IDENTIFIER=rmt23,FOR=myuserid

LPR 'TESTFILE.PRINT' (PRINTER testprt1 AT lpdsrvr1
```

The LPRM command can remove all jobs from the printer's queue, or just a specific job number, or a specific job name, as shown in Example 5-8.

Example 5-8 LPRM samples

```
LPRM (PRINTER testprt1 AT lpdsrvr1
LPRM 273 (PRINTER testprt1 AT lpdsrvr1
LPRM myjob (PRINTER testprt1 AT lpdsrvr1
```

The LPRSET command can specify the print server's DNS name or its IP address, or query the current default setting, or display a confirmation of the setting, as shown in Example 5-9.

Example 5-9 LPRSET samples

```
LPRSET (QUERY
EZB1020I Your LPR printer is currently set to <unassigned> at <unassigned>.

LPRSET testprt1@lpdsrvr1.ibm.com (TYPE
EZB1023I Printer set to testprt1 at lpdsrvr1.ibm.com.

LPRSET (QUERY
EZB1020I Your LPR printer is currently set to testprt1 at lpdsrvr1.ibm.com.

LPRSET testprt1@10.1.1.20
LPRSET (QUERY
Your LPR printer is currently set to testprt1 at 10.1.1.20.
```

If LPRSET is used to set the default printer and server, then the other commands do not have to specify the printer or server, as shown in Example 5-10.

Example 5-10 Abbreviated LPR commands use the default setting

```
LPQ
LPR 'TESTFILE.PRINT'
LPRM 273
LPRM myjob
```

5.3 Infoprint Server

This section provides an overview of IP printing with the Infoprint Server. The Infoprint Server is a program product for the z/OS platform that uses z/OS UNIX System Services. This product is the basis for a total print serving solution for the z/OS environment. It enables you to consolidate your print workload from many servers onto a central z/OS print server.

This section discusses the following topics:

- Description of the Infoprint Server
- Configuration of Infoprint Server

5.3.1 Description of the Infoprint Server

The Infoprint Server executes in the z/OS UNIX System Services environment and requires z/OS UNIX system planning efforts and system setup.

The Infoprint Server provides support for LAN and host printing using your z/OS system as a centralized print management platform for the printing needs of the entire enterprise. It works together with data stream transforms that other IBM products provide. Figure 5-3 shows how most of the components of Infoprint Server fit into your z/OS system. The components of Infoprint Server and the transform products are shaded. The Infoprint Server Transforms components include:

- z/OS UNIX Shell Transform commands
- Infoprint Server Transforms
- Coax support

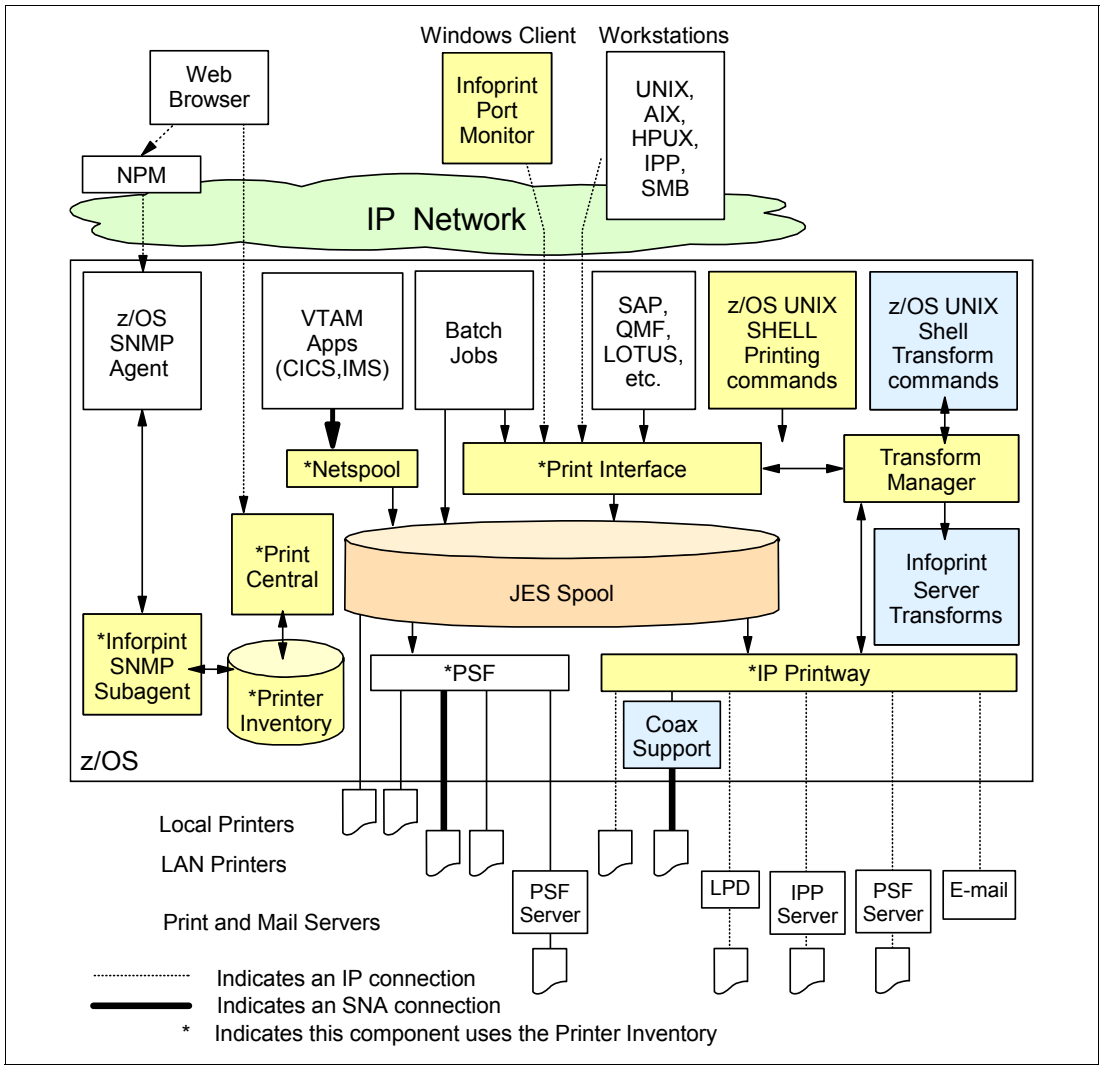


Figure 5-3 Infoprint components

Components of the Infoprint Server

The components of the Infoprint Server are described in this section.

Printer Inventory and Printer Inventory Manager

The Printer Inventory Manager controls the Printer Inventory. The Printer Inventory consists of files in the z/OS UNIX file system (zFS) that contain information about each printer and email destination. The Printer Inventory also contains system configuration information for IP IBM PrintWay™ and IBM Print Services Facility™ (PSF) for z/OS.

Infoprint Server Windows Client

The Infoprint Server Windows client consists of the Infoprint Port Monitor, which sends print requests and job attributes to the Print Interface.

Print Interface

The Print Interface processes print requests from remote clients and from the local z/OS system and allocates output data sets on the JES spool. The Print Interface accepts various data formats and can transform input data streams to EBCDIC line data, ASCII text data, AFP, PCL, PostScript, PDF, or other data formats that the printer accepts. A separate transform product is required for some transforms.

NetSpool

NetSpool processes print requests from VTAM applications, such as IBM CICS® and IBM IMS™, and allocates output data sets on the JES spool. NetSpool thereby enables SNA applications to print to TCP/IP printers. NetSpool accepts SCS, 3270, and binary data streams, and can transform input data streams to EBCDIC line data, PCL, PDF, AFP, or other data formats that the printer accepts. A separate transform product is required for some transforms. However, a separate transform product is not required to convert input data streams to the line or PCL formats.

IP PrintWay

IP PrintWay transmits data sets from the JES spool to printers or print servers in a TCP/IP or SNA network and to email destinations. IP PrintWay accepts various data formats and can transform input data streams to ASCII text data, PCL, PostScript, PDF, or other data formats that the printer accepts. A separate transform product is required for some transforms.

Transform Manager

The Infoprint Server Transform Manager manages transforms provided by Infoprint Server Transforms and other IBM transform products.

Infoprint Central

Infoprint Central is a web-based application that lets help desk operators work with print jobs (output data sets) on the JES spool, printers controlled by IP PrintWay extended mode or PSF, and NetSpool logical units. It also lets operators see system status and printer definitions.

Simple Network Management Protocol (SNMP) subagent

The SNMP subagent lets you use an SNMP manager to view printer characteristics and printer status for printers that PSF controls and that do not have internal SNMP agents or that are not TCP/IP-attached to PSF. IBM Network Printer Manager for the Web (NPM), which you can download at no charge from the web, enables an operator to monitor printers throughout the network from a web browser running on any workstation.

Infoprint Server Transforms and other transforms (separate products)

IBM provides products that transform data streams from one format to another. These products are separate from the Infoprint Server.

PSF for z/OS (separate product)

The Print Services Facility (PSF) prints output on IBM AFP printers. The PSF system programmer can specify printer configuration information in the Printer Inventory for PSF to use when it starts a printer.

Advantages of using the Infoprint Server

The Infoprint Server delivers improved efficiency and lower overall printing cost with the flexibility for high-volume, high-speed printing from anywhere in the network. With the Infoprint Server, you can reduce the overall cost of printing and improve the management of output resources.

Other benefits of the Infoprint Server include:

- IP PrintWay can give you fast access to TCP/IP-connected printers and to Virtual Telecommunications Access Method (VTAM)-controlled printers.
- NetSpool automatically directs VTAM application data to the Job Entry Subsystem (JES) spool without requiring application changes, enabling SNA print to be directed to IP printers.
- Infoprint Central lets help desk operators and other authorized users or job submitters work with print jobs, printers, and NetSpool logical units (LUs); display printer definitions; and check system status. Infoprint Central is a web-based print management system.
- Infoprint Server Transforms, a separate product, provides a set of data transforms that lets you convert data to and from the AFP data format.
- IBM Infoprint XML Extender for z/OS, a separate product, lets you transform Extensible Markup Language (XML) files to AFP or PDF format for printing or emailing.
- IBM Infoprint XT Extender for z/OS, a separate product, lets you transform Xerox files to AFP format for printing or emailing. The Xerox files can be line-conditioned data streams (LCDS) or metacode data streams. XT is the IBM Xerox Transform technology.

Considerations for using the Infoprint Server

Even though the Infoprint Server is a separate product, it consists of a number of optional features, each of which requires its own configuration and planning.

The optional features of the Infoprint Server have interdependencies for each other, and the skills required to implement the product and its various features will probably cross departmental boundaries. A cooperative effort is required among all departments involved to achieve a successful implementation.

5.3.2 Configuration of Infoprint Server

Because the Infoprint Server is a separate product from the z/OS Communications Server, it is not within the scope of this book to cover the complete details of an Infoprint Server configuration. We list the major steps to be considered and then refer you to the reference material for the Infoprint Server product in 5.5, “Additional information sources for IP printing” on page 261.

As part of any implementation effort, two appendixes in this book might be beneficial in planning your work:

- Environment variables are categorized by application in Appendix A, “Environment variables” on page 395.
- Sample files for each application are listed in Appendix B, “Sample files provided with TCP/IP” on page 407. Because we cannot cover the detailed implementation of the Infoprint Server in this book, see the reference material listed in 5.3, “Infoprint Server” on page 245 for implementation details.

A high-level overview of Infoprint Server implementation follows:

- Customizing the Infoprint Server components
- Operating Infoprint Server
- Administering the Infoprint Server
- Sample Infoprint Server configuration files for minimal functionality
- Sample Infoprint ISPF panels

Customizing the Infoprint Server components

This section can help you determine which Infoprint Server components you must customize to use the different functions that the Infoprint Server provides. Table 5-2 lists the functions that the Infoprint Server provides and the Infoprint Server components you need to customize to support those functions.

Table 5-2 Infoprint functions with corresponding components

Infoprint Server function	Components
Receive print requests from these sources, and allocate output data sets on the JES spool: - Clients that use LPR to LPD protocol - Clients that use Internet Printing Protocol (IPP) - Windows clients that use Server Message Block (SMB) protocol - z/OS UNIX **`lp`**, **`lpstat`**, and **`cancel`** commands - The AOPPRINT JCL procedure - Any Windows application that supports printing - Infoprint Server Application Programming Interface - Batch jobs that specify the Print Interface subsystem on a DD statement	- Printer Inventory Manager - Print Interface - Infoprint Port Monitor for Windows (optional)
Receive print requests from VTAM applications (such as CICS and IMS), and allocate output data sets on the JES spool.	- Printer Inventory Manager - NetSpool
Select output data sets from the JES spool and send data to a remote system using one of these transmission protocols: - LPR to LPD protocol - Internet Printing Protocol (IPP) - Direct sockets printing - VTAM - Email	- Printer Inventory Manager - IP PrintWay, basic or extended mode - Print Interface (required to transform data when you use the resubmit for filtering function of IP PrintWay basic mode)
Transform data from one format to another, either automatically or with a z/OS UNIX transform command: afp2pcl, afp2pdf, afp2ps, pcl2afp, ps2afp, pdf2afp, sap2afp.	- Printer Inventory Manager - Transform Manager - Infoprint Server Transforms V1.1

Infoprint Server function	Components
Use Infoprint Central for the web to work with print jobs, IP PrintWay extended mode printers, PSF printers, and NetSpool logical units.	► Printer Inventory Manager ► Infoprint Central
View printer characteristics and the status of PSF printers using an SNMP manager.	► Printer Inventory Manager ► SNMP subagent
Store PSF system information in the Printer Inventory.	► Printer Inventory Manager

Operating Infoprint Server

This section can help you determine which operational tasks you need to do to use the Infoprint Server components customized by your installation. Table 5-3 lists the components of the Infoprint Server and the associated operational tasks.

Note: All components of the Infoprint Server require that you start the Printer Inventory Manager.

Table 5-3 Infoprint components and operational tasks

Component	Tasks
Printer Inventory Manager	► Start and stop Infoprint Server daemons. ► View messages.
Print Interface	► Start and stop Infoprint Server daemons. ► Use Infoprint Central to manage print jobs. ► Work with output data sets on the JES spool. ► View messages.
NetSpool	► Start and stop Infoprint Server daemons. ► Start and stop the NetSpool task. ► Use Infoprint Central to manage print jobs and NetSpool LUs. ► Work with output data sets on the JES spool View messages.
IP PrintWay basic mode	► Start and stop IP PrintWay FSAs. ► Maintain the IP PrintWay transmission-queue. ► Start sendmail. ► Work with output data sets on the JES spool. ► View messages.
IP PrintWay extended mode	► Start and stop Infoprint Server daemons. ► Start sendmail. ► Use Infoprint Central to work with print jobs and printers. ► Work with output data sets on the JES spool. ► View messages.
Transform Manager and Infoprint transforms	► Start and stop Infoprint Server daemons. ► View messages.
Infoprint Central	► Start and stop Infoprint Server daemons. ► Use Infoprint Central.
SNMP subagent	► Start and stop Infoprint Server daemons. ► View messages.

Administering the Infoprint Server

This section can help you determine which administrative tasks you need to do to use the Infoprint Server components customized by your installation. Table 5-4 lists the components of the Infoprint Server and the associated administrative tasks.

Table 5-4 Infoprint components and administrative tasks

Component	Tasks
Printer Inventory Manager	► Plan the Printer Inventory. ► Use ISPF panels to manage the Printer Inventory. ► Use the Printer Inventory Definition Utility (PIDU) to manage the Printer Inventory.
Print Interface	► Plan printer definitions for the Print Interface.
Infoprint transforms	► Plan printer definitions for data transforms.
NetSpool	► Plan printer definitions and printer pool definitions for NetSpool Define NetSpool printer logical units (LUs) to VTAM.
IP PrintWay	► Plan printer definitions for IP PrintWay. Use SMF type 6 accounting record written by IP PrintWay.

Sample Infoprint Server configuration files for minimal functionality

The Infoprint Server works from a configuration file, as shown in Example 5-11.

Example 5-11 Basic aopd.conf with the snmp subagent implemented

```
#--------------------------------------------------------------------------
# aopd.conf - Base with snmp subagent
#--------------------------------------------------------------------------
lpd-port-number = 515
ipp-port-number = 631
base-directory  = /var/Printsrv
ascii-codepage  = ISO8859-1
ebcdic-codepage = IBM-1047
job-prefix      = PR
inventory       = AOPD
start-daemons   = { lpd }
snmp-community  = j02cmd27
```

Example 5-12 shows how to customize message control.

Example 5-12 Certain messages can be selected for syslog

```
# aopmsg.conf - Infoprint Server Message Configuration file
#--------------------------------------------------------------------------
hardcopy-messages = list
hardcopy-message-list = {AOP3614I AOP3803E}
#hardcopy-messages = all
#hardcopy-messages = none
```

Sample Infoprint ISPF panels

The Infoprint Server provides an ISPF panel interface for creating configuration files, printer definition files, and interface options to the other Infoprint functional components. For more details and descriptions see *z/OS Infoprint Server Operation and Administration*, S544-5745.

Several sample panels are shown next. Figure 5-4 shows the main ISPF panel for an IP PrintWay printer definition.

```
                    IP PrintWay Printer Definition

Printer definition name . _______________
Description . ____________________________________________________ (extend)
Location. . . ____________________________________________________ (extend)

                        Component name       Custom values
Section                 (enter to list)      (enter to customize)
Allocation          => _______________       => __
Processing          => _______________       => *
NetSpool options    => _______________       => __
NetSpool end-of-file => _______________      => __
IP PrintWay options => _______________       => __
Protocol            => _______________       => *

_ Use DEST, CLASS, and FORMS for IP PrintWay printer selection
NetSpool LU name . ________   LU classes . . __ __ __ __ __ __   (extend)
```

Figure 5-4 ISPF IP PrintWay Printer Definition panel

Figure 5-5 shows the main ISPF panel for a PSF printer definition.

```
                      PSF Printer Definition

Printer definition name . _______________
Description . ____________________________________________________ (extend)
Location. . . ____________________________________________________ (extend)

                        Component name       Custom values
Section                 (enter to list)      (enter to customize)
Allocation          => _______________       => __
Processing          => _______________       => *
NetSpool options    => _______________       => __
NetSpool end-of-file => _______________      => __

NetSpool LU name . ________   LU classes . . __ __ __ __ __ __   (extend)
```

Figure 5-5 ISPF PSF Printer Definition panel

Figure 5-6 shows the main ISPF panel for a General printer definition.

```
                     General Printer Definition

Printer definition name . _______________
Description . ____________________________________________________ (extend)
Location. . . ____________________________________________________ (extend)

                        Component name       Custom values
Section                 (enter to list)      (enter to customize)
Allocation          => _______________       => __
Processing          => _______________       => *
NetSpool options    => _______________       => __
NetSpool end-of-file => _______________      => __
IP PrintWay options => _______________       => __

NetSpool LU name . ________   LU classes . . __ __ __ __ __ __   (extend)
Spooling mode. . . _   1. Line  2. Stream
```

Figure 5-6 ISPF General Printer Definition panel

Figure 5-7 shows the ISPF panel for the NetSpool Options component.

```
                          NetSpool Options

Printer definition name . _______________


Formatting . . . . 2   1. None  2. Convert to Line  3. Convert to PCL
   Record size . . _____
   RECFM . . . . . _   1. VB  2. VBA  3. VBM

Default owner. . . _______
Embedded attributes prefix . . _________________________
```

Figure 5-7 ISPF NetSpool Options panel

Figure 5-8 shows the ISPF panel for the IP PrintWay Options section or component.

```
                        IP PrintWay Options

Retention period:
   Successful. . . . _________   Failure . . _________
Retry time . . . . . _________
Retry limit. . . . . _____

Connection timeout . 30
Response timeout . . 600
Exits:
   Begin data set. . _______   End data set. . _______   Record. . _______

Document header  . . _____________________________________________(extend)
   / Translate document header
Document trailer . . _____________________________________________(extend)
   / Translate document trailer
_ Automatic dataset grouping (extended mode)
Dataset grouping. . . . 2  1. None  2. Job  3. Concatenate job

Formatting:
   Transparent data char . 35
   Delete form feed. . . . 1  1. None  2. Leading  3. Trailing  4. Both
   Carriage control type . _  1. None  2. Machine  3. ANSI
   _ Omit line termination at EOF

Basic Mode Formatting:
   Line termination. . . . _______
   Formatting. . . . . . . _  1. None           2. Standard
                              3. Translate only  4. Use FCB
   PostScript header . . . _  1. Add            2. Ignore
                              3. Landscape      4. Always landscape
```

Figure 5-8 ISPF IP PrintWay Options panel

Figure 5-9 shows the LPR Protocol panel.

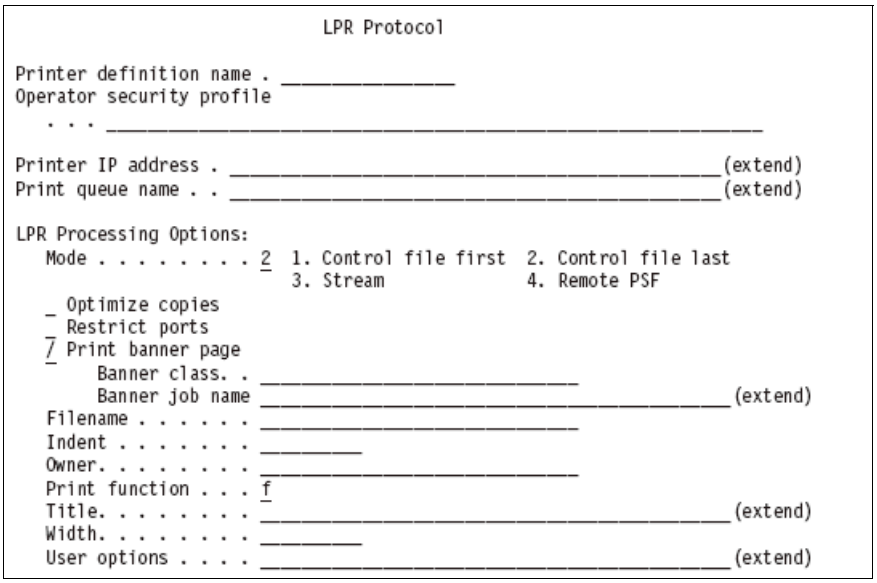


Figure 5-9 ISPF LPR Protocol panel

Figure 5-10 shows the VTAM Protocol panel.

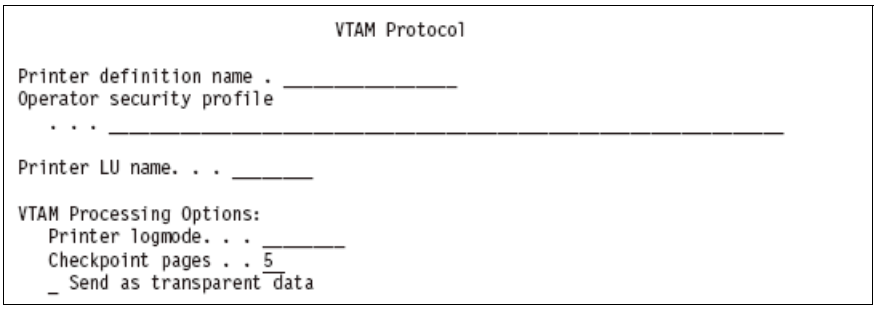


Figure 5-10 ISPF VTAM Protocol panel

Figure 5-11 shows the E-mail Protocol panel.

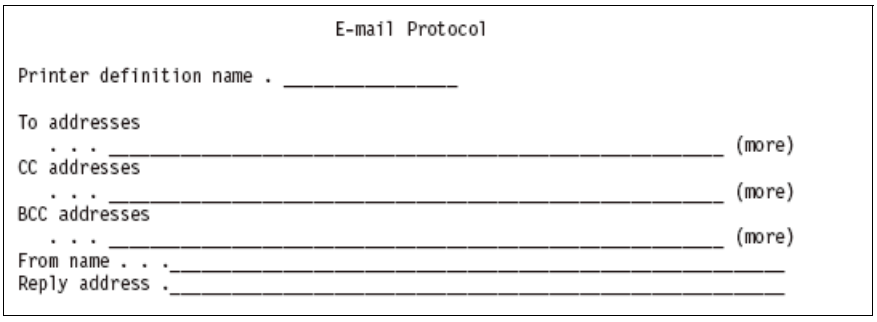


Figure 5-11 ISPF E-mail Protocol panel

Figure 5-12 shows the ISPF panel for an IP PrintWay FSS definition.

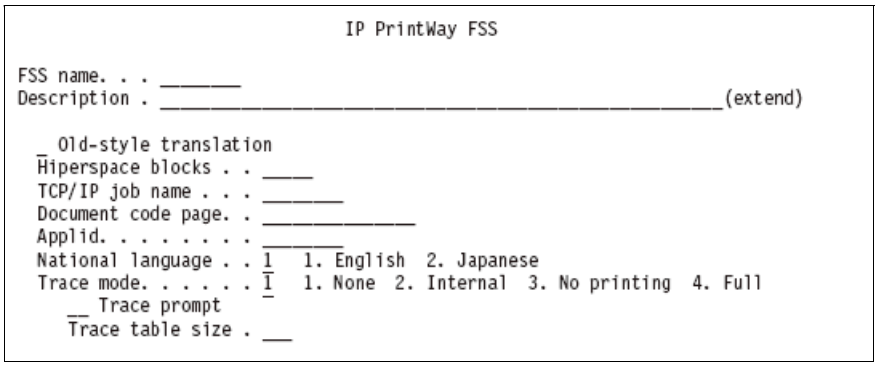


Figure 5-12 ISPF FSS printer definition panel

For complete implementation details about the Infoprint Server, see 5.3, “Infoprint Server” on page 245.

5.4 Problem determination for LPR/LPD

With z/OS Communications Server, authorization checks for many server applications are done at program startup, with an abend 4C5, reason code 77A53217, taken if the program is not properly authorized, as shown here:

```
IEA995I SYMPTOM DUMP OUTPUT
SYSTEM COMPLETION CODE=4C5  REASON CODE=77A53217
```

If an authorized program is to be started as a TSO command, it must be listed in the AUTHCMD section of IKJTSOxx, or TSO will consider it unauthorized, even if it resides in an authorized library.

See this techdoc for the details:

http://publib.boulder.ibm.com/infocenter/zos/v1r13/topic/com.ibm.zos.r13.ikjb400/usmi.htm

The LPR client command can fail if any of the following error conditions occur:

- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP might not allow the LPR client program access to the stack.
  - LPD cannot be configured or might be configured incorrectly.
- There is no LPD server listening on the remote server.
  - Ensure that you are using the correct port number.
  - Ensure that you are using the correct server name and IP address.

The TSO SMSG command provides an interactive interface to the LPD server to:

- Turn on and off diagnostics tracing.
- Query the work queue currently being used by the LPD server.

Example 5-13 shows the use of the SMSG command to turn the LPD TRACE on and off, and to show the work queue.

Example 5-13 SMSG being used to turn LPD TRACE on and off, and Print Queue

```
From the TSO ISHELL command line:
smsg lpserveb trace on
smsg lpserveb trace off
smsg lpserveb print work
```

These commands are privileged, so the commands are only accepted from users specified in the OBEY statement in the LPD server configuration data set. Responses to the SMSG command are not sent to your TSO screen. You must look in the SYSPRINT file associated with the LPSERVE job to see the responses, as shown in Example 5-14.

Example 5-14 LPSERVE SMSG command responses

```
SDSF OUTPUT DISPLAY LPSERVEB STC06560  DSID   103 LINE 10
. . .
11:19:50 EZB0831I IBM MVS LPD Version CS V1R12 on 10/01/10 at 11:19:50
11:19:50 EZB0832I
11:19:50 EZB0856I Loaded translation table from "TCPIPB.STANDARD.TCPXLBIN"
11:19:50 EZB0630I Spool allocate RC 71319616, Class  , DEST BRANCH22, ID RMT2,
OUTPUT LPD1
11:19:50 EZB0631I IKJ56875I  SYSOUT DATA SET NOT ALLOCATED, DESTINATION UNDEFINED
TO SUBSYSTEM
11:45:31 EZB0786I Command received "TRACE ON".
11:45:31 EZB0789I GetNextNote with ShouldWait of TRUE
11:49:56 EZB0790I GetNextNote returns.  Connection 0 Notification SMSG received
(8701)
11:49:56 EZB0786I Command received "TRACE OFF".
11:50:08 EZB0786I Command received "PRINT WORK".
11:50:08 EZB0731I      Work Queue start
11:50:08 EZB0733I       Work Queue end
```

On the TSO ISPF shell command line we issued the LPR command:

```
lpr 'tcpipb.tcpparms(datab30)' at 10.1.1.20 printer sysprt1 class h
```

We received the error message shown in Example 5-15 from the LPR command.

Example 5-15 Error message EZB0943I issued on the LPR command

```
EZB0943I No local printer ports available now.  Bind Conn failed.  Return Code =
-1.   Error Number = 13.  Port Number = 731.   Remote IP Addr = 10.1.1.20
 ***
```

We have RESTRICTLOWPORTS specified in the IPCONFIG section of the stack's profile member. This includes the ports that LPR wants to use. We displayed the port reservation list using the **netstat** PORTLIST command and noticed that we had forgotten to reserve the ports for LPR, as seen in Example 5-16 on page 257.

Example 5-16 Port reservation list indicates LPR ports missing (722-731)

```
d tcpip,tcpipb,n,portlist
. . .
PORT# PROT USER     FLAGS    RANGE       SAF NAME
00007 TCP  MISCSRVB DA
00009 TCP  MISCSRVB DA
00019 TCP  MISCSRVB DA
00020 TCP  OMVS
00021 TCP  OMVS     DA
00023 TCP  OMVS     DABU
      BINDSPECIFIC: 10.1.9.21
00023 TCP  TN3270B  DU
00025 TCP  SMTPB    DA
00053 TCP  NAMED9B  DA
00111 TCP  PORTMAPB DA
00512 TCP  OMVS     DABU
      BINDSPECIFIC: 10.1.9.22
00512 TCP  RXSERVEB DAU
00514 TCP  RXSERVEB DAU
00514 TCP  OMVS     DABU
      BINDSPECIFIC: 10.1.9.22
00515 TCP  LPSERVEB DA
00750 TCP  MVSKERBB DA
00751 TCP  ADM@SRVB DA
00992 TCP  TN3270B  D
00007 UDP  MISCSRVB DA
00009 UDP  MISCSRVB DA
00019 UDP  MISCSRVB DA
00053 UDP  NAMED9B  DA
00111 UDP  PORTMAPB DA
00161 UDP  SNMPDB   DA
00162 UDP  SNMPQEB  DA
00520 UDP  OMPB     D
00750 UDP  MVSKERBB DA
00751 UDP  ADM@SRVB DA
28 OF 28 RECORDS DISPLAYED
END OF THE REPORT
```

TSO users and batch jobs can issue LPR commands. Because we cannot know ahead of time what job names might be used for any TSO user ID or for batch jobs, we decided to use the wildcard approach to reserving the LPR ports. Because all of our user IDs started with CSxx, we set up an obeyfile member that would reserve the LPR ports for any job name starting with CS*, as shown in Example 5-17.

Example 5-17 OBEYFILE member showing LPR ports for LPR to be added to port reservation list

```
BROWSE    TCPIPB.TCPPARMS(LPRPORTS) - 01.00          Line 00000000 Col 001 080
PORT
 722 TCP CS*
 723 TCP CS*
 724 TCP CS*
 725 TCP CS*
 726 TCP CS*
 727 TCP CS*
 728 TCP CS*
 729 TCP CS*
 730 TCP CS*
 731 TCP CS*
```

We issued the OBEYFILE command to update the port list and then re-displayed the port reservation list to verify the LPR ports were added, as seen in Example 5-18.

Note: When adding definitions to your stack's profile using the OBEYFILE command, if you want them to be permanent, do not forget to add the statements to the actual profile source before the next recycle of the stack. If they are not added to the source by the time the stack is recycled, the statements added by OBEYFILE will be lost.

Example 5-18 Port reservation list updated with LPR ports

```
v tcpip,tcpipb,o,tcpipb.tcpparms(lprports)
 EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPB,O,TCPIPB.TCPPARMS(LPR
 PORTS)
 EZZ0300I OPENED OBEYFILE FILE 'TCPIPB.TCPPARMS(LPRPORTS)'
 EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIPB.TCPPARMS(LPRPORTS)
 '
 EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIPB.TCPPARMS(LPRPO
 RTS)'
 EZZ0053I COMMAND VARY OBEY COMPLETED SUCCESSFULLY

d tcpip,tcpipb,n,portlist
. . .
 PORT# PROT USER     FLAGS    RANGE       SAF NAME
 00515 TCP  LPSERVEB DA
 00722 TCP  CS*      DA
 00723 TCP  CS*      DA
 00724 TCP  CS*      DA
 00725 TCP  CS*      DA
 00726 TCP  CS*      DA
 00727 TCP  CS*      DA
 00728 TCP  CS*      DA
 00729 TCP  CS*      DA
 00730 TCP  CS*      DA
 00731 TCP  CS*      DA
 00750 TCP  MVSKERBB DA
 00751 TCP  ADM@SRVB DA
...
 END OF THE REPORT
```

Now we are ready to try the LPR command again. A display of the JES output queue for the LOCAL system printers shows *no output* from LPSERVEB job name *before* our LPR command, as seen in Example 5-19.

Example 5-19 JES output queue before the LPR command shows nothing from LPSERVEB

```
SDSF OUTPUT ALL CLASSES ALL FORMS     LINES 441,414    LINE 38-56 (56)
 COMMAND INPUT ===>                                        SCROLL ===> CSR
 NP   JOBNAME  JobID    Owner    Prty C Forms    Dest              Tot-Rec
      SYSLOG   STC03794 +MASTER+   96 A STD      LOCAL              95,521
      PAGTRACF JOB04094 CS08      144 H STD      LOCAL                  92
      PAGTRACF JOB04095 CS08      144 H STD      LOCAL                 230
      TRMDRACF JOB04110 CS09      144 H STD      LOCAL                 101
      CNMEUNIX STC02494 SYSPROG   144 R STD      LOCAL                  80
      TRMDSETP JOB03936 CS08      144 R STD      LOCAL                  79
      TRMDSETP JOB03937 CS08      144 R STD      LOCAL                  80
      TRMDRACF JOB03950 CS08      144 R STD      LOCAL                  21
```

```
     TRMDRACF JOB03951 CS08      144 R STD      LOCAL                    95
     TRMDRACF JOB04023 CS08      144 R STD      LOCAL                   101
 SDSF HELD OUTPUT DISPLAY ALL CLASSES  LINES 904        LINE 1-4 (4)
 COMMAND INPUT ===>                                        SCROLL ===> CSR
 NP   JOBNAME  JobID    Owner    Prty C ODisp Dest                Tot-Rec  Tot-
      CS074    JOB07178 CS07      144 H HOLD  LOCAL                    46
      CS075    JOB07180 CS07      144 H HOLD  LOCAL                    46
      CS07     TSU07131 CS07      144 S HOLD  LOCAL                   276
      CS07     TSU07132 CS07      144 S HOLD  LOCAL                   536
```

This time we decided to turn on the LPD Trace using the TSO SMSG command before we re-issued the LPR command:

```
smsg lpserveb trace on
lpr 'tcpipb.tcpparms(databa30)' at 10.1.1.20 printer sysprt1 class h
```

We can now look at SYSPRINT for LPSERVEB for the trace output, and at the SDSF JES output queue for output from LPSERVEB in class H (specified on the LPR command). The trace output was quite large, so we do not show all of it here. However, we do include certain important lines from the trace that show the progression of servicing the received print file from LPR. The trace is shown in Example 5-20.

Example 5-20 LPD trace receiving and processing a file from LPR

```
***** 10/01/10 *****
13:26:51 EZB0831I IBM MVS LPD Version CS V1R12 on 10/01/10 at 13:26:51
.................
13:27:12 EZB0834I Ready
13:30:02 EZB0786I Command received "TRACE ON".
.................
13:31:26 EZB0627I Passive open on port 515
13:31:26 EZB0782I Connection open.  Reading command.
.............
13:31:26 EZB0716I Job 998 received sysprt1 10.1.1.20
13:31:26 EZB0734I Job 998 added to work queue
13:31:26 EZB0716I Job 998 scheduled sysprt1 10.1.1.20
13:31:26 EZB0776I Released StepBlock at 00006D90
13:31:26 EZB0777I Released ConnectionBlock at 001CD000
13:31:26 EZB0824I ProcessWork starting on job queue
13:31:26 EZB0731I       Work Queue start
13:31:26 EZB0732I $        998 JOBstartPRINTING
13:31:26 EZB0733I       Work Queue end
13:31:26 EZB0825I    Job 998 for sysprt1 dispatched in state JOBstartPRINTING
13:31:26 EZB0716I Job 998 printing sysprt1 10.1.1.20
13:31:26 EZB0827I ProcessWork end with queue
13:31:26 EZB0731I       Work Queue start
13:31:26 EZB0732I $        998 JOBcontinuePRINTING
13:31:26 EZB0733I       Work Queue end
.............
13:31:26 EZB0825I    Job 998 for sysprt1 dispatched in state JOBcontinuePRINTING
13:31:26 EZB0827I ProcessWork end with queue
13:31:26 EZB0731I       Work Queue start
13:31:26 EZB0732I $        998 JOBfinishPRINTING
13:31:26 EZB0733I       Work Queue end
13:31:26 EZB0789I GetNextNote with ShouldWait of FALSE
13:31:26 EZB0824I ProcessWork starting on job queue
13:31:26 EZB0731I       Work Queue start
13:31:26 EZB0732I $        998 JOBfinishPRINTING
13:31:26 EZB0733I       Work Queue end
13:31:26 EZB0825I    Job 998 for sysprt1 dispatched in state JOBfinishPRINTING
```

```
13:31:26 EZB0716I Job 998 sent sysprt1 10.1.1.20
13:31:26 EZB0769I Job 998 removed from work queue
13:31:26 EZB0751I Released StepBlock at 00006BD8
13:31:26 EZB0716I Job 998 purged sysprt1 10.1.1.20
13:31:26 EZB0771I Released JobBlock at 000B8030
13:31:26 EZB0827I ProcessWork end with queue
13:31:26 EZB0731I       Work Queue start
13:31:26 EZB0733I       Work Queue end
13:31:26 EZB0789I GetNextNote with ShouldWait of TRUE
```

We turned the trace off and issued another LPR command, specifying a different member to be printed, this time:

```
lpr 'tcpipb.tcpparms(datab31)' at 10.1.1.20 printer sysprt1 class h
```

When the trace is off, the output in LPSERVEB's SYSPRINT appears as it does in Example 5-21.

Example 5-21 LPSERVEB normal processing messages when trace is off

```
13:58:02 EZB0716I Job 996 received sysprt1 10.1.1.20
13:58:02 EZB0716I Job 996 scheduled sysprt1 10.1.1.20
13:58:02 EZB0716I Job 996 printing sysprt1 10.1.1.20
13:58:02 EZB0716I Job 996 sent sysprt1 10.1.1.20
13:58:02 EZB0716I Job 996 purged sysprt1 10.1.1.20
```

We look at the JES output queue for output from LPSERVEB in *class H* (specified on the LPR command). We now see two new entries in the output queue for LPSERVEB that were not there before:

- One is for LPSERVEB's job 115 when the trace was on (DATAB30).
- The other is for LPSERVEB's job 88 when the trace was off (DATAB31).

These entries are seen in Example 5-22.

Example 5-22 LPSERVEB entries in the JES output queue

```
SDSF OUTPUT ALL CLASSES ALL FORMS     LINES 76        LINE 1-4 (4)
 COMMAND INPUT ===>                                      SCROLL ===> CSR
 NP   JOBNAME  JobID    Owner    Prty C Forms    Dest                Tot-Rec
      LPSERVEB   STC06154 TCPIP     144 H STD     LOCAL                  19
      LPSERVEB   STC06154 TCPIP     144 H STD     LOCAL                  19
```

If we select these two entries, we can verify that the DATAB30 and DATAB31 members are indeed in the queue, ready for printing on the local system printer, waiting in class H, as seen in Example 5-23.

Example 5-23 LPSERVEB contents of the two print files waiting to be printed

```
 SDSF OUTPUT DISPLAY LPSERVEB STC06154  DSID   111 LINE
WTSC30.ITSO.IBM.COM:TCPIPB.TCPPARMS(DATAB30)

TCPIPJOBNAME TCPIPB
HOSTNAME WTSC30B
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR 127.0.0.1
NSPORTADDR  53
RESOLVEVIA  UDP
```

```
RESOLVERTIMEOUT  10
RESOLVERUDPRETRIES 1

 SDSF OUTPUT DISPLAY LPSERVEB STC06154  DSID   112 LINE
WTSC30.ITSO.IBM.COM:TCPIPB.TCPPARMS(DATAB31)

TCPIPJOBNAME TCPIPB
HOSTNAME WTSC31B
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR 127.0.0.1
NSPORTADDR  53
RESOLVEVIA  UDP
RESOLVERTIMEOUT  10
RESOLVERUDPRETRIES 1
;LOOKUP DNS LOCAL
;OPTIONS NDOTS:1
;SOCKDEBUG
;SOCKNOTESTSTOR
;TRACE RESOLVER
;ALWAYSWTO
```

5.5 Additional information sources for IP printing

See the following sources for additional information about IP printing:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- Infoprint Server migration chapter in *z/OS Migration*, GA22-7499
- *z/OS Infoprint Server Introduction*, S544-5742
- *z/OS Infoprint Server Customization*, S544-5744
- *z/OS Infoprint Server Operation and Administration*, S544-5745
- *z/OS Infoprint Server User's Guide*, S544-5746
- *z/OS Infoprint Server Implementation*, SG24- 6234
- LPR/LPD is defined by RFC 1179

Tip: For descriptions of security considerations that affect specific servers or components, see "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

6

INETD

INETD, also known as the *Internet super daemon*, is an application that listens for connection requests on behalf of other applications. The handling the initial connection process, INETD passes the connection to the application associated with the targeted port. This chapter focuses on INETD functions that are available in the z/OS Communications Server.

This chapter discusses the following INETD topics.

Section	Topic
6.1, “Conceptual overview of INETD” on page 264	The basic concepts of INETD.
6.2, “A single INETD setup” on page 266	How to configure INETD to listen on behalf of a number of dependent applications.
6.3, “Problem determination for INETD” on page 273	Problem determination techniques for INETD.
6.4, “Additional information sources for INETD” on page 273	References to additional reading resources for INETD topics.

6.1 Conceptual overview of INETD

INETD is one of the standard applications provided with the z/OS Communications Server. It is a generic listener than can be used by any server that does not have its own listener. The relationship between INETD and its supported applications is shown in Figure 6-1.

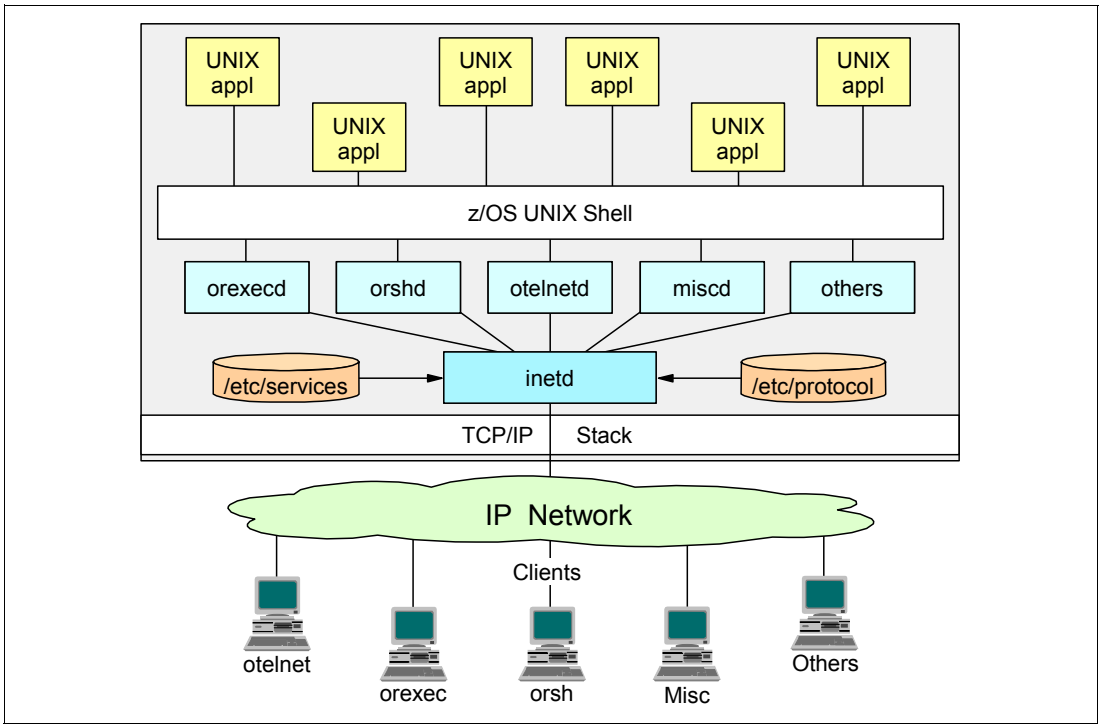


Figure 6-1 INETD and its supported applications

This section discusses the following concepts:

- What is INETD
- How does INETD work
- How can INETD be applied

6.1.1 What is INETD

The INETD servers provide access to the z/OS UNIX shell using **`otelnetd`**, **`rexecd`**, or **`rshd`**, allowing you to then run other UNIX commands and applications from there. The z/OS Communications Server includes three applications and a number of internal services that require INETD, as listed in Table 6-1.

Table 6-1 Applications that use INETD

Application	Description
z/OS UNIX Telnet server	See Chapter 8, “z/OS UNIX Telnet server” on page 315.
z/OS UNIX REXEC Server	See Chapter 9, “Remote execution” on page 325.
z/OS UNIX RSH server	See Chapter 9, “Remote execution” on page 325
sendmail and popper mail servers	See Chapter 7, “z/OS mail servers” on page 275

Application	Description
echo	Repeats any data received back to the sender.
discard	Throws away any received data.
chargen	Sends predefined or random data and discards any received data.
daytime	Sends the current date and time in user readable form.
time	Sends the current date and time in machine readable form.
popper	See Chapter 7, “z/OS mail servers” on page 275.

6.1.2 How does INETD work

INETD calls `fork()` to run in the background and disassociates itself from the controlling terminal. z/OS UNIX appends a 1 to the job name, provided the job name is less than eight characters. For example, if INETD is started with job name INETD, after the application forks the job name is INETD1. The correct job name of INETD is important to note because the TCPIP.PROFILE data set should reserve ports for INETD using the job name after INETD forks.

INETD can accept two command line parameters:

- **`-d`**
- A file name

Both parameters are optional. If **`-d`** is specified, INETD does not `fork()` at startup and all error messages are written to STDERR. If a file name is specified, then INETD uses the specified file as the configuration file instead of the default `/etc/inetd.conf`.

Note: Only one INETD daemon can be run in one MVS image. The process file that the INETD daemon uses is `/etc/inetd.pid`. You cannot change this file to another file name using either shell commands or environment variables, and it can be used only by one INETD daemon at the same time.

6.1.3 How can INETD be applied

INETD reduces system load by invoking other daemons only when they are needed and by providing several simple Internet services internally without invoking other daemons. Some applications depend on INETD to provide a listener, and those applications cannot be started without INETD.

The /etc/inetd configuration file

INETD uses a configuration file in the z/OS UNIX file system to determine for which services INETD will listen. A sample configuration file can be found in the /samples/inetd.conf file. The format of the file is:

```
<service> <socket type> <protocol> <wait/nowait> <user> <application> <arguments>
```

All parameters except the last parameter, arguments, are required on each line. Table 6-2 defines each parameter.

Table 6-2 Configuration file parameters

Parameter	Description
service	Name of the Internet service. This name must match an entry in /etc/services. By default, INETD assumes you want to listen on all IP addresses. However, you can optionally specify the service parameter in the format of IP_address:service_name to force a particular service to listen on a particular IP address.
socket type	Options are stream or dgram. Applications that use the TCP protocol are stream applications. Applications using UDP are dgram.
protocol	The IP protocol used by the application. The options are TCP, UDP, TCP4, TCP6, UDP4, or UDP6. If TCP6 or UDP6 is specified then the socket will support IPv6. You can also optionally specify the maximum receive buffer in the format of protocol,sndbuf=*n*, where *n* is the number of users.
wait/nowait	Wait indicates the server is single threaded and the application will issue an accept() API call itself and process connections one at a time. Nowait indicates the application is multi threaded. In nowait mode, INETD issues the accept() API call and passes a connected socket to the application. The application in nowait mode can process multiple connections at a time. You can also optionally specify the maximum number of simultaneous users allowed by the application by using the format nowait.*n* or wait.*n*, where *n* is the number of users.
user	The user ID the application should run under.
application	The full path to the executable file for the application INETD should start.
arguments	Up to 20 optional arguments that can be passed to the application.

6.2 A single INETD setup

INETD is a simple application with limited configuration options. INETD requires an active stack and an `/etc/inetd.conf` configuration file. We show a single design scenario for INETD. Use INETD only if you want to start **`otelnetd`**, **`orexecd`**, **`orshd`**, or if you want to use the internal services of INETD for testing purposes. In our simple design we will configure INETD with all services active. If a particular service is not desired in your environment, simply comment out the statements for the undesired service.

6.2.1 Description of the INETD setup

In this section we discuss the INETD implementation using the more common configuration options. We show a configuration file that starts all servers and internal services. This design allows the INETD configuration to be easily changed when INETD is running. Simply comment out a line in the configuration file and send a SIGHUP signal to the INETD task to stop a service.

It is never a good idea to start unnecessary server applications that will not be used. Therefore, we recommend giving careful consideration to your INETD configuration file and only allowing services to be started that will be used. For example, if you have no need for the internal services provided by INETD, then comment out the associated lines for the internal services in the configuration file.

6.2.2 Configuration tasks for INETD setup

The tasks are:

- Create an INETD configuration file
- Update Port statements in /etc/services
- Reserve ports in PROFILE.TCPIP

Create an INETD configuration file

To create an INETD configuration file:

1. First, copy the sample configuration file from /samples/inetd.conf to a new location such as `/etc/inetd.conf`.
2. Next, examine the configuration file and determine which services you want to use. Add or comment out lines as necessary. In this example, we uncomment the lines in the sample and add additional lines for each server.
3. Finally, determine under which user name INETD and the servers started by INETD will run. We recommend using OMVSKERN. The user name should have read access to the BPX.DAEMON facility class in RACF.

Example 6-1 shows our INETD configuration file.

Example 6-1 INETD configuration file

```
###
# Internet server configuration database
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 2001
# 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.
#
# /etc/inetd.conf
#
#          Internet server configuration database
#
#  $01=PYQ0049,  HOT7705, 010130, PDJP: Correct paths and remove
#        unsupported services (FIN APAR OW45915
#
# Services can be added and deleted by deleting or inserting a
# comment character (i.e. #) at the beginning of a line
#
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
pop3     stream tcp nowait OMVSKERN /usr/sbin/popper popper
echo     stream tcp nowait OMVSKERN internal
discard  stream tcp nowait OMVSKERN internal
chargen  stream tcp nowait OMVSKERN internal
daytime  stream tcp nowait OMVSKERN internal
time     stream tcp nowait OMVSKERN internal
echo     dgram  udp wait   OMVSKERN internal
discard  dgram  udp wait   OMVSKERN internal
```

```
chargen  dgram  udp wait   OMVSKERN internal
daytime  dgram  udp wait   OMVSKERN internal
time     dgram  udp wait   OMVSKERN internal
```

Update Port statements in /etc/services

For each service in the INETD configuration file, make sure a matching entry is specified in the ETC.SERVICES data set or `/etc/services` file. See our `/etc/services` file in Example 6-2.

Example 6-2 The /etc/services file

```
echo            7/tcp
echo            7/udp
discard         9/tcp
discard         9/udp
daytime         13/tcp
daytime         13/udp
chargen         19/tcp
chargen         19/udp
otelnet         23/tcp
time            37/tcp
time            37/udp
pop3            110/tcp
exec            512/tcp
login           513/tcp
shell           514/tcp
```

Reserve ports in PROFILE.TCPIP

Although not required, we suggest that you reserve ports in PROFILE.TCPIP to the INETD job name. If you do not reserve ports, it is possible that some other application will bind to the ports normally reserved for INETD servers. Our PORT statement in PROFILE.TCPIP is shown in Example 6-3 and reserves the ports for job name INETD1.

Important: Remember that the job name of INETD changes after INETD is started.

Example 6-3 PORT statement from PROFILE.TCPIP

```
PORT
    7    TCP   INETD1
    7    UDP   INETD1
    9    TCP   INETD1
    9    UDP   INETD1
    13   TCP   INETD1
    13   UDP   INETD1
    19   TCP   INETD1
    19   UDP   INETD1
    23   TCP   INETD1 BIND 10.1.9.21
    37   TCP   INETD1
    37   UDP   INETD1
    110  TCP   INETD1
    512  TCP   INETD1 BIND 10.1.9.22
    513  TCP   INETD1
    514  TCP   INETD1 BIND 10.1.9.22
```

Note: See Chapter 8, “z/OS UNIX Telnet server” on page 315, for details about why port 23 might be bound to a specific IP address.

6.2.3 Activation and verification of INETD

There are two methods for starting INETD:

- Shell script in `/etc/rc`
- As a started task

We recommend starting the INETD daemon using a shell script in `/etc/rc`, which causes INETD to be started automatically when z/OS UNIX is started.

The best method to verify that INETD is running correctly is to issue a **`netstat`** command to see the listener connections for INETD.

The activation and verification techniques for INETD and its listener processes are:

- Start INETD
- Verify INETD initial startup
- Verify the echo service
- Verify the discard service
- Verify the daytime service
- Verify the chargen service
- Verify the time service

Start INETD

A sample shell script is shown in Example 6-4. An example JCL procedure to start INETD is included in Example 6-5.

Example 6-4 Shell script commands in /etc/rc to start INETD

```
_BPX_JOBNAME='INETD'
/usr/sbin/inetd &
```

Example 6-5 JCL to start INETD

```
//INETD   PROC INETDENV=INETDENV
//INETD   EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//         PARM='PGM /usr/sbin/inetd /etc/inetd.conf'
//STDOUT   DD PATH='/tmp/inetd-stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDERR   DD PATH='/tmp/inetd-stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDENV   DD DSN=SYS1.TCPPARMS(&INETDENV.),DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=SYS1.TCPPARMS(DATA&SYSCLONE.)
```

Note: You can use the `_CEE_ENVFILE` environment variable in the PARM field of the JCL to point to a file that contains other environment variables. The file can be a UNIX file, a zFS, or a z/OS MVS data set. When it is an MVS data set, the data set must be allocated with RECFM=V.

RECFM=F must *not* be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in z/OS UNIX. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Verify INETD initial startup

Issue the TSO (NETSTAT) or UNIX (**netstat**) command, filtering on the client ID of INETD. If INETD is started correctly, the output of the **netstat** command should be similar to the output shown in Example 6-6.

```
NETSTAT CONN (CLIENT INETD*
   or
netstat -c -E INETD1
```

Example 6-6 Output of netstat after INETD is started

```
. . .
User Id  Conn     State
-------  ----     -----

INETD1   00000049 Listen
  Local Socket:   10.1.9.22..512
  Foreign Socket: 0.0.0.0..0
INETD1   00000046 Listen
  Local Socket:   0.0.0.0..19
  Foreign Socket: 0.0.0.0..0
INETD1   00000048 Listen
  Local Socket:   0.0.0.0..7
  Foreign Socket: 0.0.0.0..0
INETD1   00000045 Listen
  Local Socket:   0.0.0.0..13
  Foreign Socket: 0.0.0.0..0
INETD1   0000004C Listen
  Local Socket:   10.1.9.21..23
  Foreign Socket: 0.0.0.0..0
INETD1   00000047 Listen
  Local Socket:   0.0.0.0..9
  Foreign Socket: 0.0.0.0..0
INETD1   00000044 Listen
  Local Socket:   0.0.0.0..37
  Foreign Socket: 0.0.0.0..0
INETD1   0000004D Listen
  Local Socket:   0.0.0.0..110
  Foreign Socket: 0.0.0.0..0
INETD1   0000004B Listen
  Local Socket:   10.1.9.22..514
  Foreign Socket: 0.0.0.0..0
INETD1   0000004A Listen
  Local Socket:   0.0.0.0..513
```

```
  Foreign Socket: 0.0.0.0..0
INETD1   00000042 UDP
  Local Socket:   0.0.0.0..9
  Foreign Socket: *..*
INETD1   00000041 UDP
  Local Socket:   0.0.0.0..19
  Foreign Socket: *..*
INETD1   00000043 UDP
  Local Socket:   0.0.0.0..7
  Foreign Socket: *..*
INETD1   00000040 UDP
  Local Socket:   0.0.0.0..13
  Foreign Socket: *..*
INETD1   0000003F UDP
  Local Socket:   0.0.0.0..37
  Foreign Socket: *..*
```

Note that there are 15 entries associated with job name INETD1, which match the 15 services that were defined in the INETD configuration file. The **netstat** output only verifies that INETD is started and that it has *listeners* for each application. However, the **netstat** command does not verify that each of the servers is working.

We now verify that the internal services of INETD are working correctly by using the DOS Telnet client included in Microsoft Windows to Telnet to each port and return the output. The remote execution servers (**orexecd** and **orshd**) are discussed in detail in Chapter 9, “Remote execution” on page 325, and **otelnetd** is discussed in Chapter 8, “z/OS UNIX Telnet server” on page 315.

Verify the echo service

The echo service can be verified by using a Telnet client to Telnet to port 7 tcp, pressing Enter, and then slowly typing `This is a test`. If the echo service is working correctly, each letter is repeated as it is typed. We received the output shown in Figure 6-2.

Figure 6-2 Verification of the echo service

Verify the discard service

The discard service can be verified by using a Telnet client to Telnet to port 9 tcp, pressing Enter, and typing `This is a test`. If the echo service is working correctly, the curser moves, but each character is discarded as it is received. We received the output shown in Figure 6-3.

Figure 6-3 Verifying the discard service

Verify the daytime service

The daytime service can be verified by using a Telnet client to Telnet to port 13 tcp and pressing Enter if needed. The daytime service responds by printing the human readable date and time string and then disconnecting. We received the output shown in Figure 6-4.

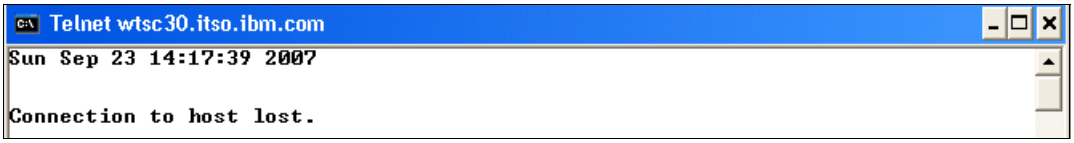


Figure 6-4 Verifying the daytime service

Verify the chargen service

The chargen service can be verified by using a Telnet client to Telnet to port 19 tcp. The chargen service sends a known string of text until the client disconnects. We received the output shown in Figure 6-5.

Figure 6-5 Verifying the chargen service

Verify the time service

The time service can be verified by using a Telnet client to Telnet to port 37 tcp. The time service sends an unreadable binary string representing the time and date. We received the output shown in Figure 6-6.

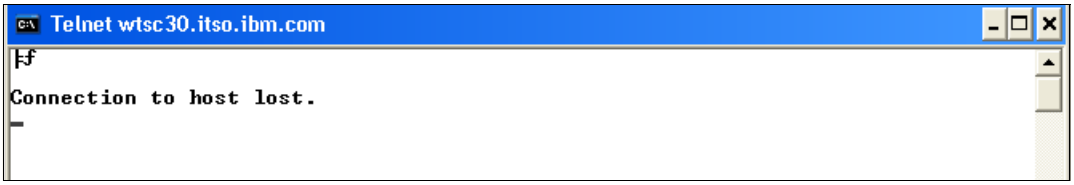


Figure 6-6 Verifying the time service

6.3 Problem determination for INETD

To perform problem determination with INETD itself or with one of the internal servers, start INETD with the **-d** option to enable debugging.

Note: The **-d** option prevents INETD from forking at startup. Therefore, the job name does not change. Because INETD will not fork in debugging mode, the behavior of INETD might also change. Additionally, the **-d** option results in numerous messages sent to the STDERR data stream.

See Chapter 9, “Remote execution” on page 325 and Chapter 8, “z/OS UNIX Telnet server” on page 315 for information regarding remote execution or **otelnetd**.

6.4 Additional information sources for INETD

See the following sources for additional information about INETD:

- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- The INETD man page (Issue **man inetd** from the z/OS UNIX shell.)

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

7

z/OS mail servers

This chapter focuses on mail services that are available in z/OS Communications Server. Basic Internet mail protocols provide mail (note) and message exchange between TCP/IP hosts. In addition, z/OS Communications Server provides facilities for the transmission of data that cannot be represented as 7-bit ASCII text. Four standard protocols can apply to this type of mail, and each protocol is recommended. The term *Simple Mail Transfer Protocol* (SMTP) is used frequently to refer to the combined set of protocols, because they are closely interrelated. However, strictly speaking, SMTP is just one of the following protocols:

- SMTP server
- z/OS Communications Server SMTP (CSSMTP)
- z/OS UNIX sendmail
- z/OS UNIX popper

This chapter discusses the following topics.

Section	Topic
7.1, “Conceptual overview of z/OS mail applications” on page 276	The basic concepts of mail servers.
7.2, “z/OS CSSMTP, a mail forwarding SMTP client” on page 278	Basic concepts of z/OS Communications Server SMTP (CSSMTP).
7.3, “z/OS SMTP as a mail server” on page 282	Commonly implemented mail server design for z/OS SMTP with configuration examples and verifications procedures.
7.4, “Using sendmail and popper as mail servers” on page 295	The sendmail and popper configuration examples and verification procedures.
7.5, “Using sendmail as a client” on page 309	The sendmail client setup configuration examples and verification procedures.
7.6, “Problem determination for the mail facilities” on page 312	Problem determination tools and commands for z/OS mail services.
7.7, “Additional information sources for mail servers” on page 314	References to additional documentation for z/OS mail services.

7.1 Conceptual overview of z/OS mail applications

Simple Mail Transfer Protocol (SMTP) and sendmail are standard applications that are provided with z/OS Communications Server, as illustrated in Figure 7-1. SMTP uses the Pascal API, which communicates directly with the TCP transport layer of the TCP/IP stack. The Pascal API bypasses communication with the z/OS UNIX logical and physical file systems. z/OS Communications Server SMTP (CSSMTP) and sendmail use Language Environment sockets.

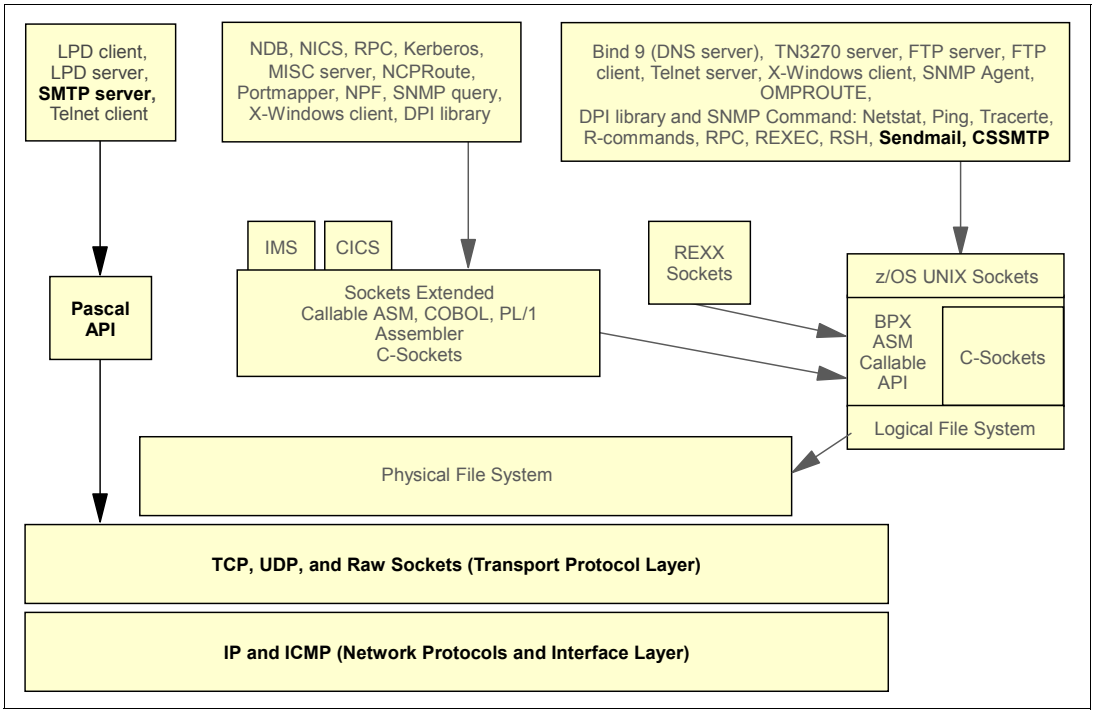


Figure 7-1 z/OS mail services

This section provides an overview of z/OS mail services and includes the following topics:

- z/OS mail services
- How z/OS mail services work
- How z/OS mail services are applied

7.1.1 z/OS mail services

z/OS supports the following mail functions:

- The z/OS Communications Server SMTP (CSSMTP) client sends mail messages from a Job Entry Subsystem (JES) spool data set to an SMTP server.
- Simple Mail Transfer Protocol Server (SMTPD) in the z/OS environment
- The sendmail program as a z/OS UNIX component
- The Post Office Protocol 3 (POP3) mail delivery agent, also known as *popper*

The various standards that define the protocols for sending electronic mail are numerous. The Request for Comments (RFC) numbers that represent these standards and their terminology might be confusing. We provide an overview of the main RFC numbers and their descriptions in the next section.

RFC 821 defines a client and server protocol. As usual, the client SMTP initiates the session (that is, sends SMTP), and the server responds (that is, receives SMTP) to the session request. Because the client SMTP frequently acts as a server for a user mailing program, it is often simpler to refer to the client as the *sender SMTP* and to the server as the *receiver SMTP*.

7.1.2 How z/OS mail services work

Figure 7-2 illustrates the relationship of the z/OS SMTP server and the client network. The users are clients who enter client commands or who request other mail services.

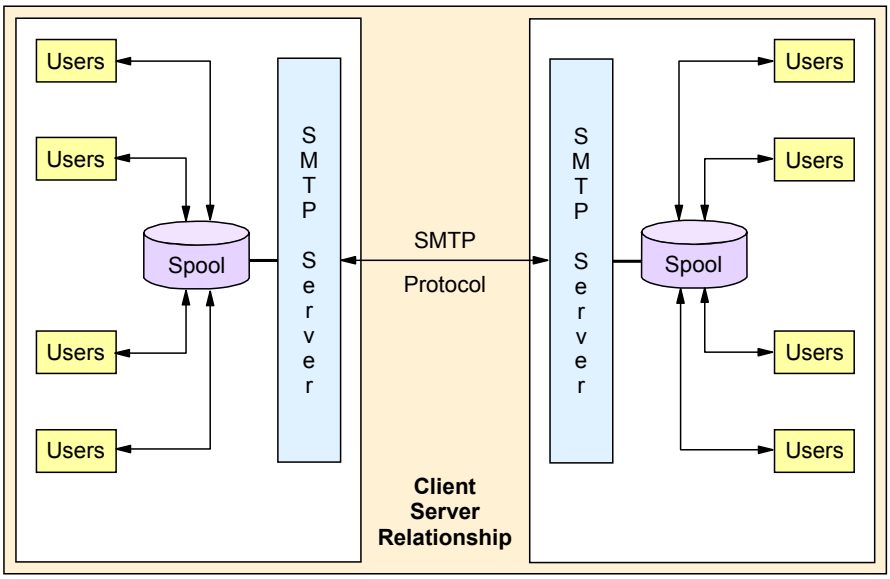


Figure 7-2 SMTP client/server relationship

Several of the more popular standards are as follows:

- A standard for the exchange of mail between two computers (STD 10/RFC 821), which specifies the protocol used to send mail between TCP/IP hosts. STD 10/RFC 821 dictates that data sent through SMTP is 7-bit ASCII data, with the high-order bit cleared to zero. This standard is SMTP itself.
- A standard (STD 11) on the format of the mail messages, contained in two RFCs:
  - RFC 822 describes the syntax of mail header fields and defines a set of header fields and their interpretation.
  - RFC 1049 describes how to use a set of document types other than plain text ASCII in the mail body, where the documents themselves are 7-bit ASCII that contain imbedded formatting information (PostScript, Scribe, SGML, TEX, TROFF, and DVI are all listed in the standard). The official protocol name for this standard is MAIL.
- A standard for the routing of mail using the Domain Name System, which is described in RFC 974. The official protocol name for this standard is DNS-MX.
- Multipurpose Internet Mail Extensions (MIME), defined in RFCs 2045 to 2049, which specify a mechanism for encoding text and binary data as 7-bit ASCII within the mail envelope defined by RFC 822.
- SMTP service extensions, which define a mechanism to extend the capabilities of SMTP beyond the limitations imposed by RFC 821.
- CSSMTP implements RFC 2821 and RFC 2822 for interacting with server MTAs, and supports additional RFCs for message size (RFC 1870) and security (RFC3207).

The list of mail RFC standards is much longer, but is out of scope for this book. For complete details about the standards and how they relate to each other, see *TCP/IP Tutorial and Technical Overview*, GG24-3376.

7.1.3 How z/OS mail services are applied

You can apply z/OS SMTP server using one of the following scenarios:

- You can set up z/OS SMTP to distribute mail on an IP network.
- z/OS SMTP can perform the role of a gateway server, transferring mail between the IP network and a local network job entry (NJE) network.
- You can configure z/OS SMTP to send unresolved non-local mail or all non-local mail to a mail transfer agent (MTA), sometimes called a *relay server*.

7.2 z/OS CSSMTP, a mail forwarding SMTP client

The z/OS Communications Server SMTP (CSSMTP) client sends mail messages from a JES spool data set to an SMTP server. In this section, we describe this client in the following topics:

- Advantages of using z/OS CSSMTP client
- Configuration tasks for the z/OS CSSMTP client
- Verification of the z/OS CSSMTP client

Figure 7-3 shows how the CSSMTP forwards mail messages from the JES spool data set.

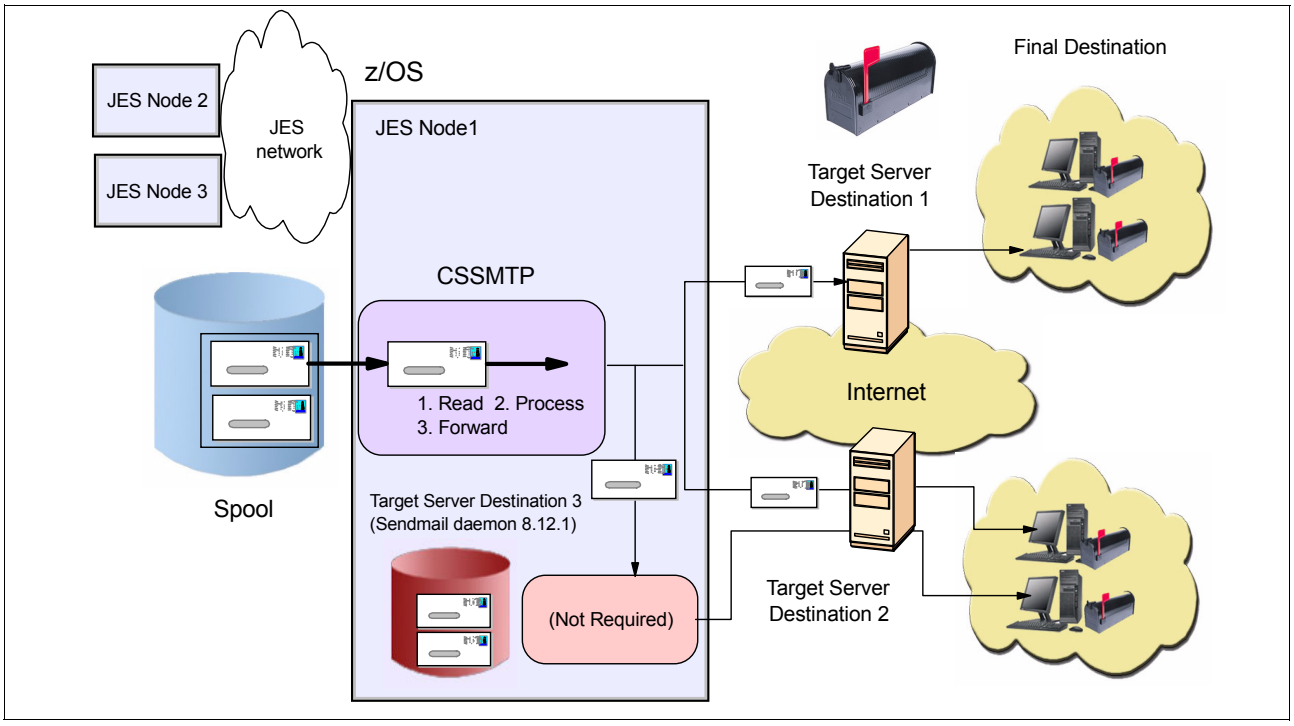


Figure 7-3 CSSMTP forwarding mail messages from a JES spool data set

7.2.1 Advantages of using z/OS CSSMTP client

In z/OS, CSSMTP provides the following advantages over SMTPD:

- CSSMTP provides the easiest and simplest solution to forwarding mail from an NJE network or from local applications that write mail to a JES spool file. If you need to receive mail from the SMTP network into your NJE network or if local TSO users use the TSO RECEIVE command to access mail, you need both SMTPD and CSSMTP. CSSMTP requires less resources than SMTP to NJE. It does not use sequential MVS data sets for all mail messages.
- CSSMTP allows existing users of SMTPD that use the forward feature to migrate easily. Thus, you can use existing batch jobs that use the old RFC 821 and RFC 822 without making changes. You must change the external writer name if CSSMTP will coexist with SMTPD.
- Use CSSMTP if you need the latest versions of the SMTP RFCs, including RFCs for message size and security. CSSMTP uses the newer mail standards RFC 2821 and RFC 2822 for interacting with server MTAs and supports additional RFCs for message size (RFC 1870) and security (RFC 3207).
- CSSMTP improves performance and storage management issues with SMTPD when forwarding mail. CSSMTP eliminates the heavy disk I/O bound nature that was a characteristics of SMTPD. CSSMTP does not require permanent disk storage for storing “active” mail messages.
- CSSMTP offers improved usability features (for example, console commands for displays, changing configuration, and logging).
- CSSMTP uses newer platform technologies and is a multitasking Language Environment application. It uses a multithreaded implementation to provide multiple JES processing threads and concurrent IP connection threads.
- CSSMTP supports both IPv4 and IPv6 addresses.
- CSSMTP provides long retry and extended retry capabilities. The long retry capability is available for up to 5 days, and is intended to compensate for short-term target server outages. For long retry, the mails are stored in-memory, and JES spools are not freed up during retries. The extended retry capability is available for a longer period or for an indefinite period, and is intended to compensate for extended target server outages. For extended retry, the mails are stored in z/OS UNIX file system. This allows for system resources such as memory and JES spool files to be freed during retries.

7.2.2 Configuration tasks for the z/OS CSSMTP client

Figure 7-4 shows a sample mail topology using CSSMTP.

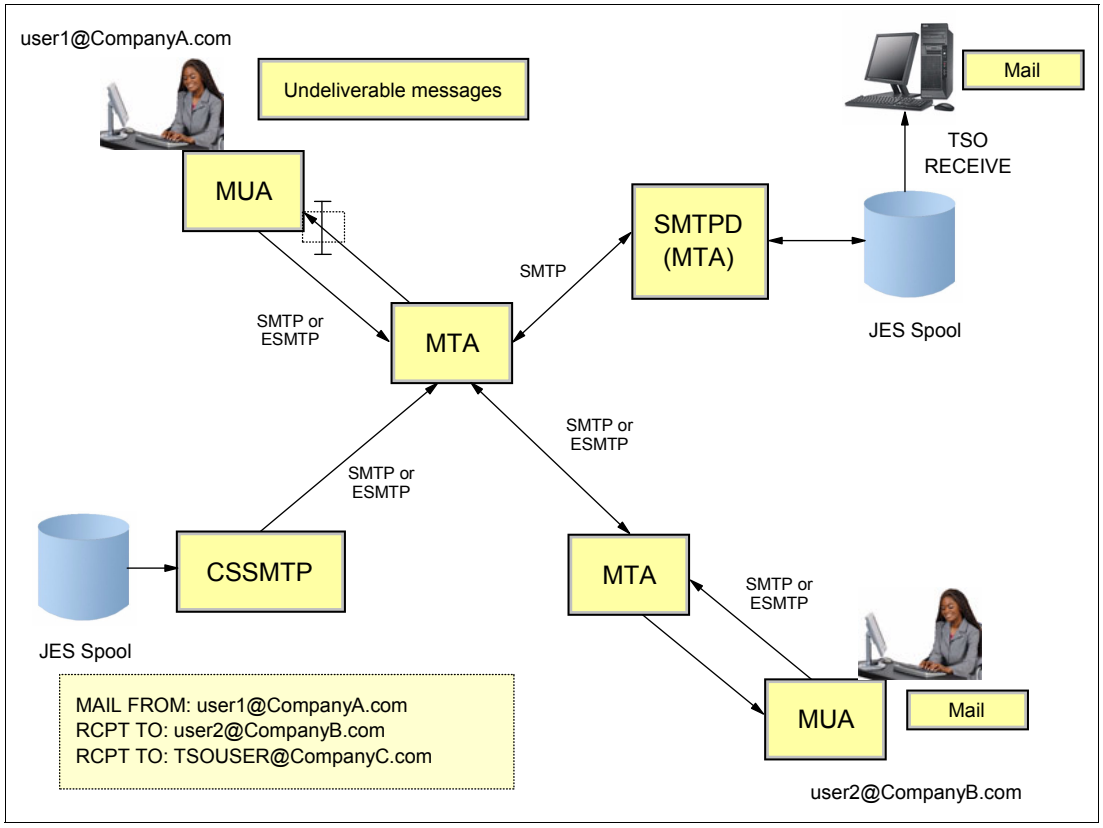


Figure 7-4 CSSMTP as a forwarder and SMTPD

To configure and start CSSMTP:

1. Customize CSSMTP procedure as follows:
   a. Copy TCPIP.SEZAINST(CSSMTP) to SYS1.PROCLIB.
   b. Create the environment variable file as follows:

```
/etc/cssmtp.env
   TZ=EST5EDT
   CSSMTP_CODEPAGE_CONFIG=IBM-1047
```

   Note: If you do not set up the time zone, the time zone is not shown in the Received header line, which is used to indicate that CSSMTP picked up a mail message. The default time value is used if the DATE header is not specified in the mail message.

   c. Create a log file named `/tmp/cssmtp.log`.
2. Optionally, define a Virtual Storage Access Method (VSAM) linear data set for the checkpoint function as shown in Example 7-1. The sample job is in SEZAINST(CSSMTPVL).

Example 7-1 Sample VSAM data set

```
DEFINE CLUSTER( +
   NAME(TCPIPA.CSSMTP.CHKPOINT)     +
   LINEAR                           +
```

```
   MEGABYTES(4 4)                  +
   VOLUME(COMST2)                  +
   SHAREOPTIONS(3 3)               +
   )                               +
  DATA(                            +
  NAME(TCPIPA.CSSMTP.CHKPOINT.DATA) +
   )
IDC0508I DATA ALLOCATION STATUS FOR VOLUME COMST2 IS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

3. Define explicit authority for all user IDs that you want to be able to start CSSMTP. Ensure that the OPERCMDS class is active and RACLISTed and that RACLIST processing is enabled. Follow these steps:
   a. Define the OPERCMDS class profile using a security product such as RACF.
   b. Grant CSSMTP access to the OPERCMDS class, and then refresh the OPERCMDS class.
4. Customize the CSSMTP configuration file as shown in Example 7-2. Set up at least one valid target server IP address using the TargetServer statement. A target server is defined as the IP address that is resolved from or configured on the TargetServer statement. To enable extended retries, code the ExtendedRetry statement. In Example 7-2, a long retry is performed every 10 minutes up to six times. After the long retry expires, the extended retry starts with 60-minute intervals for up to three days.

Example 7-2 The TargetServer statement

```
/etc/cssmtp.conf
   TargetServer
   {
      TargetIp      9.12.4.221      # target ip address
   #  TargetName    targetName      # target named to be resolved
   #  TargetMx      targetMxName    # The resolver MX query.  Only one
                                    # can be configured allowed, this is
                                    # not allowed with TargetName or
                                    # TargetIp
      ConnectPort   25              # port to connect to target server
      ConnectLimit   5              # limit the number of concurrent
                                    # connections to the target server
      MaxMsgSent    0               # when to take down a connection to
                                    # a target server and reconnect
      MessageSize   524288          # size for non-ESMTP target servers
      Secure        No              # no Transport Layer Security
   }
   RetryLimit
   {
      Count 6                       # long retry limit
      Interval 10                   # long retry interval
   }
   ExtendedRetry
   {
      Age 3                         # extended retry limit
      Interval 60                   # extended retry interval
      MailDirectory /var/cssmtp/CSSMTP/mail/   # directory to store
                                               # undelivered mails
   }
```

For information about using the CSSMTP configuration statements, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

CSSMTP configuration statements are processed during the initialization of CSSMTP or when you issue the **MODIFY procname,REFRESH** command.

5. Start CSSMTP by issuing the following command, where ***csproc*** is the CSSMTP procedure member name:

```
START CSSMPT
EZD1802I csproc INITIALIZATION COMPLETE FOR extWrtName
EZD1821I csproc ABLE TO USE TARGET SERVER ipAddress
```

7.2.3 Verification of the z/OS CSSMTP client

After following the steps described in the previous section, check the z/OS console log messages to verify that the CSSMTP client was initialized without error. If the client initialized properly, you see the following messages:

```
11.08.28 JOB04135  EZD1801I CSSMTP STARTING
11.08.28 JOB04135  EZD1840I CSSMTP UPDATED CONFIGURATION
11.08.28 JOB04135  EZD1846I CSSMTP UPDATED TARGET SERVERS
11.08.29 JOB04135  EZD1802I CSSMTP INITIALIZATION COMPLETE FOR CSSMTP
```

For more details about the CSSMTP client, *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

7.3 z/OS SMTP as a mail server

This section provides an overview of the z/OS SMTP mail server and includes the following topics:

- Description of z/OS SMTP server
- Configuration tasks for the z/OS SMTP server
- Verification of the z/OS SMTP server

7.3.1 Description of z/OS SMTP server

z/OS SMTP provides a method of originating and receiving electronic mail through the TCP/IP network using the mainframe.

Automation packages can take advantage of the SMTP functions to send alert messages to systems personnel and application administrators when batch jobs fail or critical situations arise that could adversely affect the integrity of the system environment.

A mainframe JES/NJE system that does not have a TCP/IP-capable SMTP client or server running on it can still participate in sending and receiving mail by using an SMTP gateway server on another system image.

As long as one system within a sysplex of mainframes has an SMTP server running, all sysplex members that share the sysplex's JES spool can participate in the electronic mail delivery functions.

Description of the z/OS SMTP server environments

Figure 7-5 shows the mail distribution environments of the z/OS SMTP server.

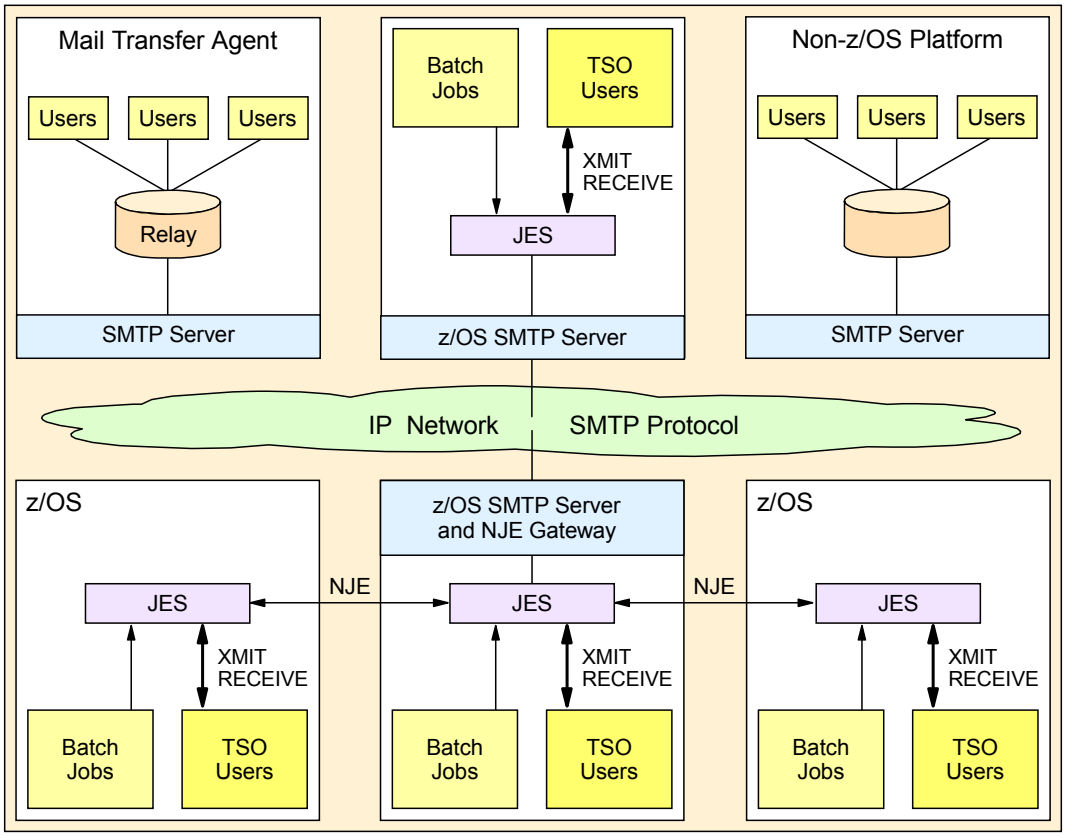


Figure 7-5 SMTP mail server: IP, NJE, relay relationships

As we introduce in 7.1.3, “How z/OS mail services are applied” on page 278, the following three scenarios are for mail services using the z/OS SMTP server:

IP network

SMTP (that is, STD 11/RFC 821) is based on end-to-end delivery. An SMTP client contacts the destination host’s SMTP server directly to deliver the mail. It keeps the mail item being transmitted until it has been successfully copied to the recipient's SMTP. This is different from the store-and-forward principle that is common in many mailing systems, where the mail item might pass through a number of intermediate hosts in the same network on its way to the destination and where successful transmission from the sender only indicates that the mail item has reached the first intermediate hop.

NJE network

This setup enables NJE users to send and receive mail to and from users on TCP networks. In various implementations, there is a possibility to exchange mail between the TCP/IP SMTP mailing system and the local JES NJE spools. This SMTP server function is a mail *gateway* or mail bridge. Sending mail through a mail gateway can alter the end-to-end delivery specification, because SMTP will only guarantee delivery to the mail-gateway host, not to the real destination host, which is located beyond the TCP/IP network. When a mail gateway is used, the SMTP end-to-end transmission is host-to-gateway, gateway-to-host, or gateway-to-gateway. The behavior beyond the gateway is not defined by SMTP.

Relay server usage — When a client originates outbound mail destined for an unknown domain (unknown to the system where the SMTP server is running), the SMTP server can optionally forward the mail to a relay server that will be able to use special SMTP DNS lookup records to determine how and where to forward the otherwise undeliverable mail. In addition, the SMTP server can be configured to forward all *out of domain* outbound mail to a relay server.

Dependencies of the z/OS SMTP server

Dependencies of the z/OS SMTP server include:

- The SMTP server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E is issued and the server terminates.
- SMTP requires access to the JES spool. It uses JES utilities to create, read, write, and purge data sets from the JES spool. JES exit programs might interfere with SMTP functions.
- The system security server, such as RACF, must have the SMTP started task name defined and authorized for use with the JES spool.
- JES parameters must be set up in a way that mail can be sent to SMTP and that local mail can be placed on the JES spool for local users.
- SMTP can be a large user of DASD. Multiple files are created for each mail note processed. Even though these files are temporary and are deleted after a note is delivered, DASD volume management is necessary to avoid contention for resources with other applications running on the system.
- DASD management packages should be run only when SMTP is down.
- SMTP requires special translation tables. The ASCII LineFeed character (X'0A') needs to be translated to an EBCDIC LineFeed (X'25'). Make sure that the proper translate table is available to the SMTP server address space.
- The use of a relay server requires a connection over the IP network, having permission to access the relay server and to send forwarded mail to it. There must be special DNS records, called Mail Exchange (MX) records, on the DNS servers in order for the SMTP server to determine the next hop, or the next relay server in the forwarding path.

Considerations for using the z/OS SMTP server

If you have specified PROFILE NOINTERCOM in your TSO user ID's profile you do not receive some SMTP server messages.

The TSO interactive interface of SMTP is command line prompt driven, and you must know the format of the subcommands and specific syntax to be followed for the data content.

The batch interface assumes a data set has been created, containing SMTP mail commands, and submitted to the JES spool where SMTP reads the commands and processes them without interactive input or control from the originating user.

After mail is delivered to a relay server, that mail is consider delivered by the SMTP server that forwarded the mail. It then becomes the responsibility of the relay server. If the relay server cannot be contacted, then the SMTP server must have a method and a procedure for storing and managing undeliverable mail. The sender must be notified, and attempts to retry delivery must be made until successful delivery is achieved, or until retry attempts are exhausted.

7.3.2 Configuration tasks for the z/OS SMTP server

There are common steps to set up the SMTP server regardless of how it is to be used or what role it plays. There are some optional steps to perform when the server is to perform special roles like that of an NJE gateway server, or when it will be forwarding mail to a relay server. The tasks to configure the z/OS SMTP server are:

- Update the TCP/IP profile configuration data set
- Update RACF to define the SMTP started task
- Customize the SMTP server procedure JCL
- Create the SMTP server configuration data set
- Customize the SMTPNOTE CLIST for TSO support
- Customize VMCF and TNF
- Customize the SYS1.PARMLIB(IKJTSOxx) member
- Determine whether NJE Gateway support is necessary: (NJE)
- Enable SMTP domain name resolution
- Define a relay server if one is planned
- Create an SMTP user exit to define and filter spam mail, if necessary

Update the TCP/IP profile configuration data set

The AUTOLOG and PORT statements should be updated to indicate the action and support that the stack needs to provide for the SMTP server. AUTOLOG indicates whether the stack should initially start the SMTP started task. PORT provides a port reservation for the port number that the SMTP server listens on. The default is port 25. If your SMTP server is to listen on a different port (other than port 25), specify the port number on the PORT statement in the server's configuration file.

Update RACF to define the SMTP started task

Every started task must be assigned a user ID, and that user ID must be granted authority to access the required resources used by the started task. This discussion assumes RACF is the security subsystem being used. If another security product is used, see its manuals for equivalent setup instructions. Before SMTP can be started, security for the procedure name and its associated user ID must be defined. Review the sample file SEZAINST(EZARACF) that contains sample security statements for this effort.

The procedure name must be added to the RACF STARTED class and have a user ID associated with it as follows:

```
RDEFINE STARTED SMTP*.* STDATA(USER(SMTP))
SETROPTS RACLIST(STARTED) REFRESH
```

Coding the started task name using the wildcard format enables us to run multiple SMTP started tasks without needing to define each one separately. Their names would all be spelled SMTP*x*, where *x* is the qualifier. They can all be assigned to the same user ID.

Use a new or existing superuser ID to associate with the job name by adding a user ID to RACF and altering it (or an existing ID) to superuser status as follows:

```
ADDUSER SMTP
ALTUSER SMTP OMVS(UID(0) PROGRAM ('/bin/sh') HOME('/'))
```

In this example the user ID name is SMTP, but any name can be used. These two RACF commands can be combined into one command by putting the OMVS parameter on the ADDUSER command line. The add and alter commands are shown separately in case the user ID already exists. If the add fails, the alter still succeeds.

If setting up a superuser ID is not desirable, you can **permit** the user ID to the BPX.SUPERUSER class using the following steps:

1. Add the user to RACF:

   ```
   ADDUSER SMTP
   ```

2. Permit the user ID:

   a. Create a BPX.SUPERUSER FACILITY class profile, if it does not already exist:

      ```
      RDEFINE FACILITY BPX.SUPERUSER
      ```

   b. If this is the first class profile, activate the FACILITY class:

      ```
      SETROPTS CLASSACT(FACILITY)
      SETROPTS RACLIST(FACILITY)
      ```

   c. Permit the user to the class:

      ```
      ALTUSER SMTP OMVS(UID(25) PROGRAM ('/bin/sh') HOME('/'))
      PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SMTP) ACCESS(READ)
      ```

      In this example, the user ID is SMTP and the UID is 25. The UID can be any nonzero number. UID 25 was used to match the well-known SMTP port number.

   d. Refresh the FACILITY class:

      ```
      SETROPTS RACLIST(FACILITY) REFRESH
      ```

Customize the SMTP server procedure JCL

A sample of the procedure is in `hlq.SEZAINST(SMTPPROC)`. You can customize data set names to meet standards of your installation. In addition, you need to complete the follow-on steps that we describe in the remaining sections, which might require that you to add one or more DD statements to this procedure. Store your updated procedure in a system procedure library, and make certain its name matches the name you put on the AUTOLOG and PORT statements in the TCP/IP profile configuration data set.

Example 7-3 shows the SMTP server procedure that we used in our testing.

Example 7-3 SMTP server Proc JCL

```
//SMTPB PROC MODULE=SMTP,DEBUG=,PARMS='NOSPIE/',SYSERR=SYSERR
//*
//* Turn on SMSG support
//*
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SMTPB EXEC PGM=MVPMAIN,
//             PARM='&MODULE,PARM=&DEBUG,ERRFILE(&SYSERR),&PARMS',
//             REGION=0M,TIME=1440
//*STEPLIB  DD DSN=SYS1.SEZATCP,DISP=SHR
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//LOGFILE  DD SYSOUT=*
//SMTPNJE  DD DSN=TCPIP.SMTPNJE.HOSTINFO,DISP=SHR
//CONFIG   DD DSN=TCPIP.TCPPARMS(SMTPB&SYSCLONE.),DISP=SHR
```

```
//*SECTABLE DD DSN=TCPIP.SMTP.SECTABLE,DISP=SHR
//*SMTPRULE DD DSN=TCPIP.SMTP.RULE,DISP=SHR
//SYSTCPD  DD DSN=TCPIP.TCPPARMS(DATAB&SYSCLONE.),DISP=SHR
```

Create the SMTP server configuration data set

A sample SMTP configuration member is in `hlq.SEZAINST(SMTPCONF)`. Use this configuration member to set up a standard SMTP server. When you are comfortable with the basic definitions, you can add optional functions such as the NJE gateway and relay server statements. For statement syntax, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For a discussion of statement use, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Example 7-4 shows unique parameters for this server instance.

Note: Defining a RACF valid user ID to the `BadSpoolField` parameter is required for SMTP customization. However, with the redesign of SMTP, using the SMTP start procedure owner's user ID produces the following message:

```
EZA5165E Invalid BadSpoolFileId Userid: SMTP
```

Previously, when SMTP detected a bad spool file, it put the bad spool file back on the JES spool with a destination of the user ID defined in `BadSpoolField`. If this user ID was itself, it caused a mail loop. (SMTP took the file back off the queue, because it was sent to itself, and kept re-processing it.) To prevent this error, the SMTP now includes a test to see whether `BadSpoolField` is itself. If it is, it issues the EZA5165E message. So now `BadSpoolField` can be any valid user ID *except* the owner of the start procedure.

Example 7-4 SMTP server configuration data set: unique parameters

```
;*********************************************************************
;  Defaults that normally aren't changed:
;*********************************************************************
PORT 25
BADSPOOLFILEID CSO7
LOG
INACTIVE 180
FINISHOPEN 120
RETRYAGE 3
WARNINGAGE 1
RETRYINT 20
MAXMAILBYTES 524288
RESOLVERRETRYINT 20
RCPTRESPONSEDELAY 60
TEMPERRORRETRIES 0
SPOOLPOLLINTERVAL 30
TIMEZONE SYSTZ
;*********************************************************************
;  INSTALLATION SPECIFIC STATEMENTS
;*********************************************************************
NJENODENAME WTSCPLX5
ALTTCPHOSTNAME SC31MAIL
MAILFILEDSPREFIX SMTPB
MAILFILEUNIT SYSDA
POSTMASTER CSO7
SMSGAUTHLIST
    SMTP
    CSO7
```

```
    CSO1
ENDSMSGAUTHLIST
;*********************************************************************
; CONFIGURATION FOR A TYPICAL NJE TO TCP/IP MAIL GATEWAY.
;*********************************************************************
GATEWAY                ; ACCEPT MAIL FROM AND DELIVER MAIL TO NJE HOSTS
NJEDOMAIN SCNJENET     ; ANY TWO NAMES WE WANT.  OTHERS MUST USE THESE
ALTNJEDOMAIN SCNJENET  ; USERO2%SCNJENET@SC31MAIL.RALEIGH.IBM.COM
NJEFORMAT PUNCH        ; NJE RECIPIENTS RECEIVE MAIL IN PUNCH FORMAT
NJECLASS A             ; SPOOL CLASS FOR MAIL DELIVERED BY SMTP TO THE
LOCALFORMAT NETDATA    ; LOCAL RECIPIENTS GET MAIL IN NETDATA FORMAT
;                      ; NETDATA ALLOWS TSO RECEIVE TO BE USED WITH MAIL
LOCALCLASS A           ; SPOOL CLASS FOR LOCAL MAIL DELIVERED BY SMTP
REWRITE822HEADER YES NOPRINT
;*********************************************************************
; The RESTRICT statement specifies addresses of users who cannot
; utilize SMTP services.
;*********************************************************************
RESTRICT RETURN          ; Return mail from restricted users
  cs09@us.ibm.com ; DON'T ACCEPT ANY MAIL FROM PRINCE CHARMING
  cs09@SCNJENET          ; VIA NJE OR TCP NETWORK.
  cs09@*                ;THIS LINE TAKES THE PLACE OF PREVIOUS 2 LINES
 *@badsite              ;DON'T ACCEPT MAIL FROM ANYONE AT HOST CASTLE
ENDRESTRICT
;*********************************************************************
;      SEND ALL NON-LOCAL MAIL TO AN MTA RELAY SERVER
;*********************************************************************
IPMAILERADDRESS 10.12.4.221
RESOLVERUSAGE YES
```

Customize the SMTPNOTE CLIST for TSO support

If you plan to support TSO users in creating outbound mail, set up the SMTPNOTE CLIST. There is a sample in `hlq.SEZAINST(SMTPNOTE)`. It is a command-line interface that prompts for required information to allow a TSO user to set up a note and have it delivered to its destination. The clist collects the necessary information from the user and builds a file that is then written to the JES spool for the SMTP server. The server discovers the note on the spool and processes the SMTP command within the file.

Users can also use IEBGENER in batch to generate a file that contains all the same SMTP commands necessary to send a note through SMTP to the IP or NJE network.

Whether the SMTPNOTE clist or the batch method, or both, will be used values for the following items must be established and then assigned:

DDNAME — This is the temporary data set used to build the SMTP commands for the note. SMTPNOTE refers to this data set as the INPUT data set. After the commands are built and the user indicates the note is ready for delivery, SMTPNOTE transmits the contents of this file (which are SMTP commands) to the JES spool destined for the SMTP server.

TEMPDSN — This is the name of the data set allocated for the DDNAME above. Make certain the data set name ends with the low-level qualifier of .TEXT and do not fully qualify the name. By not fully qualifying the name, the clist prefixes the name with the user's TSO *user ID*.

HOSTNAME — This is usually the NJE node name of the system on which the SMTPNOTE clist runs.

SMTPNODE This is the NJE node name of the system where the SMTP server runs. If the SMTP server is acting as an NJE gateway server, then it can be on a different NJE node than where the SMTPNOTE clist is running. SMTPNODE must always point to the node of the SMTP server.

SMTPJOB This is the started task job name of the SMTP server itself. The SMTPNOTE clist uses TSO XMIT to send the note to the SMTP server. XMIT uses JES facilities and the JES spool. The started task name must not be the same as any node name defined to JES. It cannot begin with the characters R, RM, or RMT because JES could get confused and think that the file should be delivered to a JES Remote instead of the SMTP server. The XMIT command sends the note file to the JES spool using the two values you specify for "*SMTPNODE.SMTPJOB*". That spool location must not be processed by any other service than the SMTP address space.

TIMEZONE This is for the system on which this SMTPNOTE clist runs. The value of SYSTZ will allow the code to dynamically retrieve the value of the time zone from the system Communication Vector Table (CVT) control block.

ATSIGN This specifies a single-byte representation of the at symbol (@) for foreign languages.

DOMAIN Some SMTP MTAs need a fully qualified name as an email address for the originator of the mail. If DOMAIN is set, then this string is appended to the HOSTNAME variable string provided in this CLIST, and the resulting fully qualified name string is *hostname.domain*. The resulting string is later used by the CLIST to create the SMTP MAIL FROM: command and the RFC 822 From: header in the mail message. The CLIST does not check validity of the content of the string. This variable should be set when sending mail to CSSMTP.

For use of the SMTPNOTE clist, see *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780.

Customize VMCF and TNF

The VMCF and TNF is required for starting SMTP, for detail information about VMCF and TNF customization, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996.

Customize the SYS1.PARMLIB(IKJTSOxx) member

The SMTPNOTE clist uses the TSO XMIT command. The XMIT command uses JES facilities to accomplish the transfer to spool. The TRANSREC statement must contain the correct node name. As an alternative, the NODESMF parameter can be coded as NODESMF((*,*)). *,* specifies that the node name is to be retrieved dynamically from JES. This specification is recommended because it eliminates the need to specify static values for node name and smfid. For more information about the TRANSREC statement, see *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

Determine whether NJE Gateway support is necessary: (NJE)

If you transmit messages to the JESSPOOL destined for the SMTP server using TSO XMIT or using IEBGENER, then you need to implement the SMTP server as a local NJE gateway. See Figure 7-6 for a list of the NJE Gateway related statements.

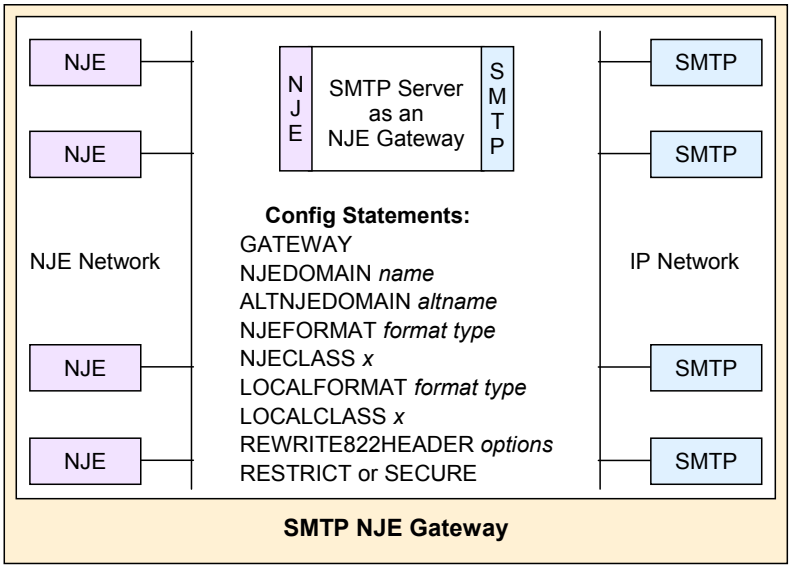


Figure 7-6 SMTP server as an NJE gateway server

If you intend to use the SMTP server as a gateway to the NJE network, be sure to consider the following tasks:

- Customize the SMTP mail headers, if necessary: (NJE).

  Usually, the default header rules supplied by IBM are sufficient for most NJE traffic. However, if you have a special condition that is not covered by the default rules, see the comprehensive discussion of customizing mail headers in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

- Set up the TCP-to-NJE gateway function: (NJE).

  Assign values for and plan to use each of the following statements in the SMTP configuration data set:

  - GATEWAY indicates that this SMTP server is an NJE gateway.
  - NJENODENAME is the node name of the local JES NJE system.
  - NJEDOMAIN sets the domain name of the NJE network to what you want it to be.
  - ALTNJEDOMAIN is an alternate domain name of the NJE network (synonym).
  - NJECLASS is the JES spool data set class for mail delivered on the NJE network.
  - NJEFORMAT is the JES spool data set format to be used.

- Plan to use the NJENODENAME statement within the SMTP configuration file.

  Do not plan to depend on the setting of the VMCF node name value either in SYS1.PARMLIB(IEFSSNxx) or set by the SYS1.PROCLIB(EZAZSSI) procedure. These values for node name or system name could be and usually are different from the NJE node name. SMTP requires you to know the actual NJE node name value. Do it the easy way by specifying it with the NJENODENAME statement.

Note: The NJENODENAME statement, if specified, must precede any of the following statements in the SMTP Configuration Data Set:

- ALTNJEDOMAIN
- MAILER
- NJEDOMAIN
- SMSGAUTHLIST

- Create the required NJE host table data set named *hlq*.SMTPNJE.HOSTINFO.

  SMTP cannot accept mail destined to a node name that JES does not have defined. The SMTP gateway server requires this data set to determine which NJE nodes are defined to JES and are thus permitted to participate in mail transfers. To build this data set you must run the following TSO command and point it to the JES initialization data set member that contains the JES node definitions:

  ```
  TSO SMTPNJE 'SYS1.PARMLIB(JES2PARM)'
  ```

  The command scans the member for node definitions in the HASPPARM DD statement of JES2 start procedure and builds the required file in *userid*.SMPTNJE.HOSTINFO, you can change the name to your consistent HLQ.

  For JES3, add the parameter at the end of the command to point to JES3, as the following line shows:

  ```
  TSO SMTPNJE data.set.pds.name(member) (JES3
  ```

  The default is JES2.

- Add the required //SMTPNJE DD statement to the SMTP server procedure JCL, pointing it to the HOSTINFO data set just created:

  ```
  //SMTPNJE DD DSN=hlq.SMTPNJE.HOSTINFO,DISP=SHR
  ```

- Determine whether you want to define a SECURE NJE Gateway.

  A SECURE statement can be added to the SMTP server configuration data set to define the server as a secure gateway between the TCP/IP network and the NJE network. When the server is operating in secure mode, only those NJE addresses in the SMTP security table are allowed to use the mail services of this server. The SMTP server rejects mail to or from an unauthorized user. An unauthorized user is one whose user ID is *not* in the table. This table is coded as records in the required data set:

  ```
  mailfiledsprefix.SMTP.SECTABLE with LRECL=255 and RECFM=VB
  ```

  In addition, it is pointed to by adding the required DD statement to the SMTP server proc:

  ```
  //SECTABLE DD DSN=mailfiledsprefix.SMTP.SECTABLE,DISP=SHR.
  ```

  The *mailfiledsprefix* hlq is also defined within the SMTP configuration data set.

  You must create a second required data set. It is used when NJE mail is rejected. Its contents are used as a memo note that is sent to the unauthorized user whose mail is rejected. The name of this data set is as follows:

  ```
  mailfiledsprefix.SECURITY.MEMO with LRECL=255 and RECFM=VB
  ```

  It is allocated dynamically by the SMTP server when needed. You do *not* add a DD statement to the SMTP server procedure JCL.

  For examples and details about the format and syntax of the records for both data sets, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

- Determine whether you want to define a RESTRICT NJE Gateway.

  A RESTRICT statement can be added to the SMTP server configuration data set to indicate those user IDs who are not allowed to use the mail services of this server. You code a list of user IDs within the RESTRICT list statement. For details about the syntax of the RESTRICT statement you must see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Note: RESTRICT and SECURE are both optional settings. However, they are mutually exclusive.

Figure 7-7 shows NJE-related parameters.

```
NJENODENAME WTSCPLX5
...
SMSGAUTHLIST
    SMTP
    CSO7
    CSO1
ENDSMSGAUTHLIST
...
;*********************************************************************
; CONFIGURATION FOR A TYPICAL NJE TO TCP/IP MAIL GATEWAY.
;*********************************************************************
GATEWAY                ; ACCEPT MAIL FROM AND DELIVER MAIL TO NJE HOSTS
NJEDOMAIN SCNJENET     ; ANY TWO NAMES WE WANT.  OTHERS MUST USE THESE
ALTNJEDOMAIN SCNJENET  ; USERO2%SCNJENET@SC31MAIL.RALEIGH.IBM.COM
NJEFORMAT PUNCH        ; NJE RECIPIENTS RECEIVE MAIL IN PUNCH FORMAT
NJECLASS A             ; SPOOL CLASS FOR MAIL DELIVERED BY SMTP TO THE
LOCALFORMAT NETDATA    ; LOCAL RECIPIENTS GET MAIL IN NETDATA FORMAT
;                      ; NETDATA ALLOWS TSO RECEIVE TO BE USED WITH MAIL
LOCALCLASS A           ; SPOOL CLASS FOR LOCAL MAIL DELIVERED BY SMTP
REWRITE822HEADER YES NOPRINT
```

Figure 7-7 SMTP server configuration data set: NJE parameters

Figure 7-8 shows restrict and secure related settings.

```
;*******************************************************************
; 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
;*******************************************************************
; The RESTRICT statement specifies addresses of users who cannot
; utilize SMTP services.
;*******************************************************************
RESTRICT RETURN          ; Return mail from restricted users
  cs09@us.ibm.com        ; DON'T ACCEPT ANY MAIL FROM PRINCE CHARMING
  cs09@SCNJENET          ; VIA NJE OR TCP NETWORK.
  cs09@*                ;THIS LINE TAKES THE PLACE OF PREVIOUS 2 LINES
 *@badsite              ;DON'T ACCEPT MAIL FROM ANYONE AT HOST CASTLE
ENDRESTRICT
```

Figure 7-8 SMTP server configuration data set: secure and restrict parameters

Enable SMTP domain name resolution

The SMTP server configuration statement RESOLVERUSAGE indicates whether domain name resolution is to be performed. If name resolution is desired, make certain that the //SYSTCPD DD statement is in the SMTP server procedure JCL and that it points to a valid TCPDATA file containing correct DNS server information. If name resolution is not desired, you must code RESOLVERUSAGE NO. For a complete discussion about how DNS services are used by the SMTP server and how it processes DNS MX type records, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Define a relay server if one is planned

Non-local mail must go through an MTA to get to another host. SMTP supports the following configuration statements to assist in forwarding non-local mail:

- IPMAILERNAME, for non-local mail destined for SMTP servers in the IP network using a host name
- IPMAILERADDRESS, for non-local mail destined for SMTP servers in the IP network using a static IP address
- MAILER, for non-local mail destined for SMTP servers in the NJE network using the JES spool

There is a special situation where you might want to send *all* non-local mail to a relay server, and not just *unresolved* non-local mail. For a detailed discussion about how to direct all non-local mail to a relay server, see the topic on sending non-local messages to other mail servers in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Figure 7-9 shows relay server related parameters.

```
;*********************************************************************
;      SEND ALL NON-LOCAL MAIL TO AN MTA RELAY SERVER
;*********************************************************************
IPMAILERADDRESS 10.12.4.221
RESOLVERUSAGE YES
```

Figure 7-9 SMTP server configuration data set: relay server parameters

Create an SMTP user exit to define and filter spam mail, if necessary

There is a sample user exit in `hlq.SEZAINST(SMTPEXIT)`. For details about implementing and managing the exit, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

7.3.3 Verification of the z/OS SMTP server

To receive a detailed trace on how SMTP is resolving a particular host name, you can issue the TSO SMSG SMTP TRACE command in TSO, or the MVS MODIFY (F) command, or use a SYSTCPT DD statement in the SMTP cataloged procedure. Samples of these methods are:

```
TSO SMSG SMTPB TRACE
F SMTPB,SMSG,TRACE
//SYSTCPT DD SYSOUT=*
```

Note: The TSO SMSG command works when issued from TSO only and should not be issued from the operator console. SMSG SMTP is not supported in batch. It uses VMCF and the PASCAL interface to queue information and will not print the information to a DD card. Issue the MVS MODIFY (F) command from the MVS console or an application console interface such as that provided within SDSF or NetView.

An automation tool could issue a MODIFY command and then notify an administrator when a large amount of mail is queued for SMTP, based on the output of the command when using the NUMQueue parameter.

You can also add the TRACE RESOLVER statement when configuring the TCPIP.DATA data set, but this will trace name resolution for all the applications using the name server. To prevent the console log from becoming too large, only use the TRACE RESOLVER statement for debugging.

TIMEZONE parameter

To verify that the value of SYSTZ set to the TIMEZONE parameter in a SMTP server configuration works correctly, we submitted a batch SMTP. Example 7-5 shows messages in a spool file created after we submitted a batch job. We can see -0400 (**1**) instead of the printable zone name such as EST. (The `0400` represents HH:MM.)

Example 7-5 SMTP Spool file

```
Received: from WTSC31B.ITSO.IBM.COM by WTSC31B.ITSO.IBM.COM (IBM MVS SMTP CS)
   with BSMTP id JOB03510; Mon, 24 Sep 07 15:02:34 -0400 1
Date: Mon, 24 Sep 07 15:02:34 -0400 1
From: <CS10@WTSC31B.ITSO.IBM.COM>

THIS IS A TEST MAIL.
```

7.4 Using sendmail and popper as mail servers

In this section we establish sendmail and popper as mail servers. We discuss the following topics:

- Description of sendmail and popper
- Configuration tasks for sendmail and popper
- Verification of sendmail and popper setup

7.4.1 Description of sendmail and popper

The sendmail application is an industry-standard mail application that is widely used on the Internet. The sendmail application that is included with the z/OS Communications Server is based on sendmail Version 8.12.1. Popper is a Post Office Protocol 3 (POP3) mail delivery agent. It allows a remote user to use a remote mail user agent (MUA) to read, compose, and manage email that has been delivered to z/OS. See Figure 7-5 on page 283 to review the role of a mail server.

The sendmail and popper applications are full-featured industry standard mail applications. The sendmail application is also capable of acting as a client that can be used to send email. The sendmail application supports MIME attachments and also has built-in security features such as TLS/SSL sockets.

Note: Because of the complexity of sendmail, we highly recommend that you become familiar with the industry-accepted publication about sendmail: *sendmail, 4th Edition* by Costales, Assmann, Jansen, Shapiro, from O'Reilly Media, Inc. (ISBN 10: 0-596-51029-2).

The following topics are discussed in this section:

- Dependencies for sendmail and popper
- Considerations when using sendmail and popper
- MTA, MUA, and MDA
- M4 preprocessor
- Alias file
- The statistics file
- The help file
- The queue directory
- The sendmail configuration file
- Popper

Dependencies for sendmail and popper

Sendmail needs information that, for example, defines the local mailer program, defines which file system is responsible for deferred delivery, and defines which file contains information about alias names and real names. The only required file is the sendmail configuration file (usually `/etc/sendmail.cf`). However, that file can list other directories and files that must be created before sendmail can start. To start sendmail using the example configuration file, you must create a configuration file, a mail queue directory, the `/etc/mail` directory, and a local-host-names file.

Considerations when using sendmail and popper

Using sendmail can be complex. The sendmail program is incapable of interfacing with JES. If you want to submit batch job email, then you should use SMTP rather than sendmail.

Note: The sendmail program is hard-coded to use ISO8859-1 and IBM-1047 encoding. It does not use DBCS character sets at all. If you want to use DBCS character sets, use MIME.

MTA, MUA, and MDA

When discussing sendmail and popper, other books refer to *mail transfer agents* (MTAs), *mail user agents* (MUAs), and *mail delivery agents* (MDAs):

- An MTA is any application that sends a prepared email message to a remote MTA. Sendmail, Microsoft Exchange, and IBM Lotus® IBM Domino® are examples of MTAs.
- An MUA is any application that allows a user to read, compose, and manage email. IBM Lotus Notes®, Microsoft Outlook, and Netscape Navigator are all applications that have MUA capabilities.
- An MDA is any application that sits between an MUA and MTA and manages delivering a message from an MTA to an MUA. A MDA is an optional piece of an email system. Popper is an example of an MDA.

In z/OS UNIX, sendmail is an MTA, **/bin/mail** is a command-line MUA, and popper is an MDA. In our example configuration, we use sendmail as an MTA, Microsoft Outlook Express as an MUA, and popper as an MDA.

M4 preprocessor

The m4 macro processor is a front-end processor for many programming languages. Besides replacing one string of text with another, the m4 macro processor provides the following features:

- Arithmetic capabilities
- File manipulation
- Conditional macro expansion
- String and substring functions

The m4 macro preprocessor can be given input that will generate a z/OS UNIX sendmail configuration file. It takes as input a user-defined master configuration source file (.mc file) defines mail delivery mechanisms using files provided in the samples directory. When you run the m4 preprocessor, you need files that contain input definitions and an output file that will be your sendmail configuration file. The input files are:

`/m4/cf.m4` Provides support for different include files such as `cfhead.m4` and `proto.m4`.

`/cf/sample.mc` References other files and their locations. The sample.mc file is located in the directory `/usr/lpp/tcpip/samples/sendmail/cf`.

You can use the `cf.m4` and `sample.mc` files to create your own `sendmail.cf` output file or you can use the pre-generated sample configuration file. In our environment, we first used the sample configuration file to get sendmail started, and then later used m4 to generate a new configuration file.

Alias file

The alias file is used to convert an alias name to another recipient's name. A large number of names could potentially be listed in an alias file. To efficiently deal with a potentially large alias file, sendmail uses an alias database file created from a alias file. The database version of the

alias file significantly improves lookup speed. A database file is created from the alias file with the **/usr/sbin/newaliases** or **/usr/sbin/sendmail -bi** commands.

The statistics file

The `sendmail.st` statistics file is used by sendmail to record the number and sizes of all incoming and outgoing mail messages handled by each delivery agent. Statistics are kept for each of the following delivery agents:

- Local delivery agent
- SMTP delivery agent
- UUCP delivery agent

Statistical information is collected if option (O) is defined in the `sendmail.cf` file. Sendmail does not create the statistics file; you must manually create the file first, before using the statistics option. You can view the statistics file using either of the following commands:

- `mailstats -C </etc/sendmail.cf>`
- `mailstats -s </etc/sendmail.st>`

Use of the statistics file is optional.

The help file

The `sendmail.hf` file is the help file implemented for SMTP (and ESMTP). You will find this file in the /usr/lpp/tcpip/lib directory. If you want to view this file, you can issue the following command:

```
obrowse /usr/lpp/tcpip/lib/sendmail.hf
```

Use of the help file is optional.

The queue directory

Sendmail creates a queue directory the first time sendmail is run. The location of the queue directory is specified in the sendmail configuration file. If a mail message cannot be transmitted sendmail stores it in the queue directory until the message can be transmitted successfully. Possible reasons for queuing a message include:

- The remote machine is down.
- There are temporary disk problems.
- Sendmail or another MTA on the other machine is not started.
- There are TCP communication problems.

If you have the permission to look at the queue directory, you might find it empty, which implies that all messages have been sent. Alternatively, you might find some `dfxxxxxxxx` and `qfxxxxxxxx` files, which are messages that are waiting to be sent. When a message is queued, it is split into two parts. Each part is saved in a separate file. Header information is contained in `qf` files. The actual body of the message in included in the df files. You can use the **obrowse** command to read these files. Superuser permission is normally needed.

The `qf` files include the following header lines:

1. Version of the qf file: V2 means above the V8.8 sendmail version.
2. Time created in seconds to limit the message to remain in the queue.
3. Determines the time to wait before retrying delivery.
4. Number of attempts for each delivery.
5. Priority when processed from the queue.

6. After a system crash the message is stored in lost+found under the referenced number in case the `qf` file lost its directory entry.
7. Reason why the message was stored in the queue (= deferred).
8. Full canonical name of the sender's machine.
9. Sender's address.

   For security reasons, this is the real recipient of the message.
10. Recipient's address.
11. Header information for the return path in case the message cannot be delivered.
12. HReceived: where the message came from.
13. Header information when the message was received by the MTA.
14. Header information about the sender.
15. Header information about the full name of the sender.

All header information is based on entries in sendmail's configuration file `/etc/sendmail.cf`. You can also get queue information for all messages in the queue by running the sendmail command with the **-bp** command-line switch (printing the queue).

The sendmail program offers two different methods for processing the queue:

- Process the queue periodically with the **-q** and **-h** command-line switches.
- Process the queue once with only the **-q** command-line switch and then exit.

The sendmail configuration file

The sendmail configuration file is read and parsed by the sendmail program every time sendmail starts. It lists the locations of important files and specifies the default parameter values to be used within the sendmail program. The parameters within the configuration file define sendmail's behavior and contain rules and rule sets for tasks, such as rewriting the mailing address. The configuration file uses a basic syntax, which consists of a command followed by a value.

Examples of configuration commands are as follows:

M Define a mail delivery agent.
R Define rewriting rules.
H Define a header.
P Define delivery priorities.
T Define a trusted user.
D Define a macro.
O Define an option.
S Declare a rule-set start.

There are many more definitions in the `sendmail.cf` file, so this file can be complex. If you browse through the configuration file found in `/usr/lpp/tcpip/samples/sendmail/cf/sample.cf`, you will find a complex example. To make it easier to start sendmail, you only need to copy the `/usr/lpp/tcpip/samples/sendmail/cf/sample.cf` file to `/etc/sendmail.cf`. The sample sendmail file was used in our tests.

The `sample.cf` file we used was created automatically by running the m4 macro preprocessor with the master input file in `/usr/lpp/tcpip/samples/sendmail/cf/sample.mc`. If you browse through this `sample.mc` file, you notice that the most common information is already defined, which provides a base to start with if you later want to add to the sample `sendmail.cf` file.

Popper

The only configuration required for popper is to update `/etc/services` and `/etc/inetd.conf`. Using popper is discussed in detail in 7.4.3, “Verification of sendmail and popper setup” on page 304.

7.4.2 Configuration tasks for sendmail and popper

In this section we use the samples provided with the z/OS Communications Server to start sendmail. We then start popper and use a third-party MUA to retrieve email from z/OS.

Important: The following tasks must be performed by the superuser ID, UID=0. If you use another user ID, then sendmail can issue the following messages:

```
EZZ9927I Permission denied (real uid not trusted).
or
dbm map "Alias0": unsafe map file /etc/mail/aliases: EDC5111I Permission
denied.
WARNING: cannot open alias database /etc/mail/aliases
```

If your ID is not a superuser ID and you forget to issue an **`su`** command before you create the `/etc/mail` directory, you can issue the command **`chown 0:0 /etc/mail`** to change the ownership of the `/etc/mail` directory and bypass these errors.

The following tasks must be performed to start sendmail:

- Create the /etc/mail directory
- Create the configuration file
- Create the queue directory
- Create the local-host-names file
- Create the popper maildrop directory
- Create the aliases database
- Update /etc/inetd.conf for popper
- Update /etc/services
- Update PROFILE.TCPIP
- Create sendmail start procedure
- Start inetd
- Start sendmail

Create the /etc/mail directory

From the z/OS UNIX System Services shell, issue the following command:

```
mkdir /etc/mail
```

Both the `/etc` directory and the `/etc/mail` subdirectory should have file permissions of 755. If they need to be changed, issue the commands:

```
chmod 755 /etc
chmod 755 /etc/mail
```

Create the configuration file

The sample configuration file for sendmail is in the following location:

```
/usr/lpp/tcpip/samples/sendmail/cf/sample.cf.
```

So, to help make testing sendmail easier, we can do both with IPv4 or only with IPv4:

- Both with IPv4 and IPv6 support

  If your environment is already customized to support AF_INET6 in BPXPRMxx parmlib and the affinitive TCP/IP stack is also customized for IPv6, then you can simply copy the `sample.cf` and use this copy. From the z/OS UNIX System Services shell, issue the following command:

  ```
  cp /usr/lpp/tcpip/samples/sendmail/cf/sample.cf /etc/mail/sendmail.cf
  ```

- Only with IPv4 support

If your environment is not customized to support AF_INET6, or the TCP/IP stack is not customized for IPv6, perform the following steps to create your own `sendmail.cf`:

1. Retrieve the m4 preprocessor.

   Retrieve the m4 macro preprocessor from the z/OS Toys and Tools web page at:

   `http://www-03.ibm.com/systems/z/os/zos/features/unix/bpxa1ty1.html`

   Download it and FTP to the z/OS UNIX file system with binary, then place it in `/tmp/m4.bin.pax.Z`. Issue the following command to unpax the m4 macro preprocessor:

   ```
   pax -rzf /tmp/m4.bin.pax.Z
   ```

   Then the m4 preprocessor is created in /tmp/bin/m4 with its information file in `/tmp/info/m4.info`.

2. Create the `.mc` file.

   Copy the example `.mc` file from `/usr/lpp/tcpip/samples/sendmail/sample.mc` using the following command:

   ```
   cp /usr/lpp/tcpip/samples/sendmail/cf/sample.mc /etc/mail/sendmail.mc
   ```

   Comment out the IPv6 statement from the `/etc/mail/sendmail.mc` file using the **`oedit /etc/mail/sendmail.mc`** command, as shown in Example 7-6.

Example 7-6 /etc/mail/sendmail.mc

```
divert(-1)
#
#  Sample configuration for z/OS
#
#
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM
# 5694-A01
# (c) Copyright IBM Corp. 1992, 2005
#
divert(0)dnl
VERSIONID(`z/OS sample configuration  2007/09/20')
OSTYPE(zOS)dnl
DOMAIN(generic)dnl
divert(-1)
#
# Set the name of the required zOS configuration file.
# A sample is shipped in /usr/lpp/tcpip/samples/sendmail/cf/zOS.cf
#
# Remove the undefine statement for confZOS_FILE to specify the
# zOS.cf file
divert(0)dnl
define(`confZOS_FILE', `/etc/mail/zOS.cf')dnl
undefine(`confZOS_FILE')dnl
```

```
divert(-1)
#
# Add WorkAroundBrokenAAA for IPv6 errors on name servers
divert(0)dnl
define(`confBIND_OPTS', `WorkAroundBrokenAAAA')dnl
divert(-1)
#
# listen for both inet and inet6 sockets
divert(0)dnl
DAEMON_OPTIONS(`Name=MTA,   Family=inet ')dnl
#DAEMON_OPTIONS(`Name=MTA-6, Family=inet6')dnl
divert(-1)
#
# define mailers
divert(0)dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

3. Build the configuration file.

   To build the configuration file, go to the directory containing `m4/cf.m4`, and then use the m4 macro preprocessor with the following commands:

```
$ cd /usr/lpp/tcpip/samples/sendmail/cf
$ /tmp/bin/m4 ../m4/cf.m4 /tmp/sendmail.mc > /etc/mail/sendmail.cf
```

Create the queue directory

From the z/OS UNIX System Services shell, issue the following command:

```
mkdir /usr/spool/mqueue
```

Create the local-host-names file

The local-host-names file identifies host names for which sendmail is to receive mail. We will keep this file empty for now, but it must be created in order for sendmail to start.

From the z/OS UNIX System Services shell, issue the following command:

```
touch /etc/mail/local-host-names
```

Create the popper maildrop directory

From the z/OS UNIX System Services shell, issue the following command:

```
mkdir /usr/mail/popper
chmod 777 /usr/mail/popper
```

Create the aliases database

The alias file maps names in email addresses to local user accounts. You must list in the alias file all local users who are to receive email. Our alias file is shown in Example 7-7.

Example 7-7 Contents of our /etc/mail/aliases file

```
MAILER-DAEMON:IBMUSER
postmaster:IBMUSER
cs01:CS01
cs07:CS07
nobody: /dev/null
```

You can create the file with the command **oedit /etc/mail/aliases**. Then change the file permissions with **chmod 600 /etc/mail/aliases**. Finally, run the **/usr/sbin/sendmail -bi -f /etc/mail/sendmail.cf** command to create a binary database. The message in Example 7-8 shows when you successfully create the aliases database.

Example 7-8 Messages for aliases database

```
EZZ9988I /SC31/etc/mail/aliases : 5 aliases, longest 9 bytes, 68 bytes total
```

Important: You cannot set the group write or other write bits on in the mode field of the aliases file. Otherwise, message EZZ9993I will be issued with the 265 053B006C error code for the Group writable file.

Update /etc/inetd.conf for popper

Popper is started by inetd and requires an entry in the `/etc/inetd.conf` file. The statement is shown in Example 7-9. If you have already started INETD as described in Chapter 6, “INETD” on page 263, then you have already completed this step.

Example 7-9 The /etc/inetd.conf statement for popper

```
pop3       stream tcp nowait bpxroot     /usr/sbin/popper popper
```

Update /etc/services

/etc/services should contain the lines shown in Example 7-10. If you have already started INETD as described in Chapter 6, “INETD” on page 263, then you have already completed this step.

Example 7-10 The /etc/services entries

```
smtp            25/tcp          mail
pop3           110/tcp          popper
```

If the lines are not present, then add them.

Update PROFILE.TCPIP

We need to reserve TCP port 25 for sendmail. The easiest way to accomplish this is to reserve port 25 TCP to omvs. Because popper uses port 110 TCP, and is started by INETD, we need to reserve port 110 to INETD1. Example 7-11 shows our PROFILE.TCPIP statements.

Example 7-11 PORT statements in PROFILE.TCPIP

```
25 TCP OMVS
110 TCP INETD1
```

Create sendmail start procedure

To manage the sendmail flexibly, we create a z/OS start procedure SENDMAIL with BPXBATCH utility as Example 7-12 shows.

Example 7-12 SYS1.PROCLIB(SENDMAIL)

```
//SENDMAIL PROC SMAILENV=SMAILENV
//SENDMAIL EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//   PARM='PGM /usr/sbin/sendmail -bd -q5m -C /etc/mail/sendmail.cf &'
//STDOUT   DD PATH='/tmp/sendmail-stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
```

```
//          PATHMODE=SIRWXU
//STDERR   DD PATH='/tmp/sendmail-stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDENV   DD DSN=TCPIP.TCPPARMS(&SMAILENV.),DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.TCPPARMS(DATAB&SYSCLONE.)
```

Use the STDENV DD statement in the start procedure to point to a data set that specifies the environment variables for sendmail, as Example 7-13 shows.

Example 7-13 TCPIP.TCPPARMS(SMAILENV)

```
_BPX_JOBNAME=SENDMAIL
_BPXK_SETIBMOPT_TRANSPORT=TCPIP
```

Define the RACF profile SENDMAIL.* in the STARTED class with a superuser ID as an owner in its STDATA:

```
ADDGROUP SNDMGRP OMVS(GID(26))
ADDUSER SENDMAIL DFLTGRP(SNDMGRP) NOPASSWORD OMVS(UID(0) HOME('/'))
RDEFINE STARTED SENDMAIL.* STDATA(USER(SENDMAIL))
SETROPTS RACLIST(STARTED) REFRESH
PERMIT BPX.DAEMON CLASS(FACILITY) ID(SENDMAIL) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Start inetd

If you have already started INETD as described in Chapter 6, "INETD" on page 263, then you have already completed this step. Otherwise, issue the system command **s inetd** or the following shell commands from the z/OS UNIX System:

```
export _BPX_JOBNAME=INETD
export _BPXK_SETIBMOPT_TRANSPORT=TCPIP
/usr/sbin/inetd &
```

Note: If you already have an INETD running and it does not have the popper customized, then after you add pop3 statements into its configuration files, you need to recycle the INETD. To stop the INETD, issue the following command in the z/OS UNIX System Services shell:

```
kill -TERM `cat /etc/inetd.pid`
```

Start sendmail

To start the sendmail, issue the system command **s sendmail**. The BPXBATCH utility passes the shell command in the PARM= field to OMVS with its environment variables in the STDENV DD statement. It is the same as issuing the following command in the z/OS UNIX System Services shell:

```
export _BPX_JOBNAME=SENDMAIL
export _BPXK_SETIBMOPT_TRANSPORT=TCPIP
/usr/sbin/sendmail -bd -q5m -C /etc/mail/sendmail.cf &
```

This method of starting sendmail leaves a running process that handles mail and processes any queued mail once every hour.

To stop the sendmail, issue the following command in the z/OS UNIX System Services shell:

```
kill -TERM `cat /etc/mail/sendmail.pid`
```

7.4.3 Verification of sendmail and popper setup

As shown in Example 7-14, you can use the **ps** command to verify that sendmail has started.

Example 7-14 Output of ps command

```
50462948 ttyp0000  0:00 /usr/sbin/sendmail
```

Next, as shown in Example 7-15, you can use **netstat** to determine whether sendmail has put a listener up on port 25 TCP.

Example 7-15 Output of netstat

```
SENDMAIL 0000003E Listen
  Local Socket:   0.0.0.0..25
  Foreign Socket: 0.0.0.0..0
```

Finally, use the freely available email client Mozilla Thunderbird as an MUA to check the email.

We obtained Thunderbird 2 from the following site:

http://www.mozilla.com

We installed it on a PC. While installing Thunderbird, we were prompted to set up an email account.

The following figures show the account settings that are necessary to allow Thunderbird to access sendmail and popper running on the z/OS system. The first panel of the configuration wizard in Mozilla Thunderbird asks what type of account is being set up. We are setting up an email account, as shown in Figure 7-10.

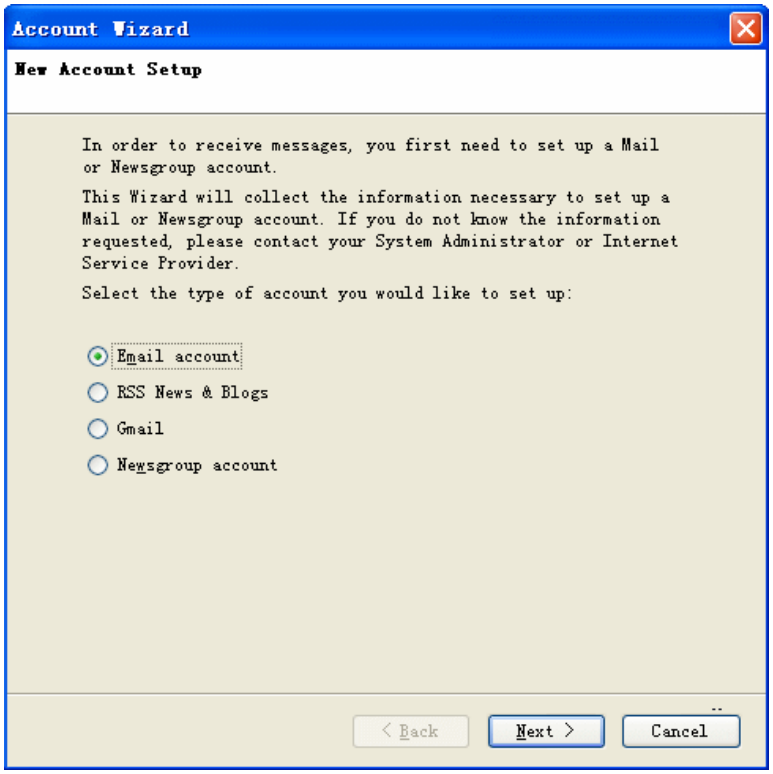


Figure 7-10 First panel of the Mozilla Thunderbird account wizard

The second panel in the account wizard asks for the name and email address, as shown in Figure 7-11.

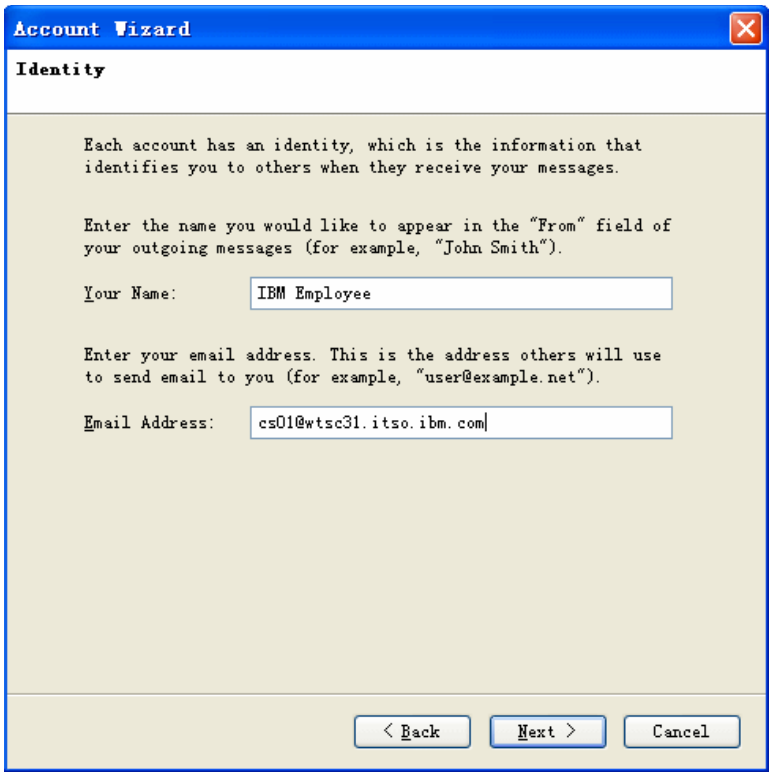


Figure 7-11 Second panel of the Mozilla Thunderbird account wizard

The third panel in the account wizard asks for the type of incoming server, that server's host name, and the outgoing server host name. Popper is the application responsible for transferring incoming mail and it uses the POP3 protocol, so we select a server type of POP. The incoming and outgoing servers are the same in our case because we are running popper and sendmail on the same LPAR. Our settings for the third panel are shown in Figure 7-12.

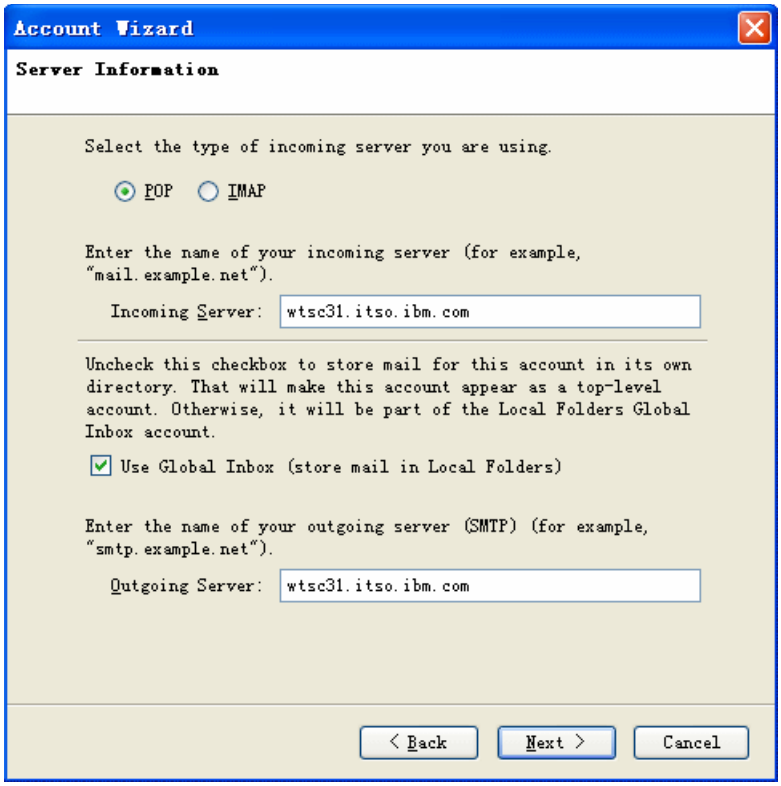


Figure 7-12 Third panel of the Mozilla Thunderbird account wizard

The fourth panel in the account wizard asks for the user ID to present to the incoming and outgoing servers. Again, because we are using the same LPAR for both popper and sendmail, there is only one user ID. Our user ID of cs02 is shown in Figure 7-13.

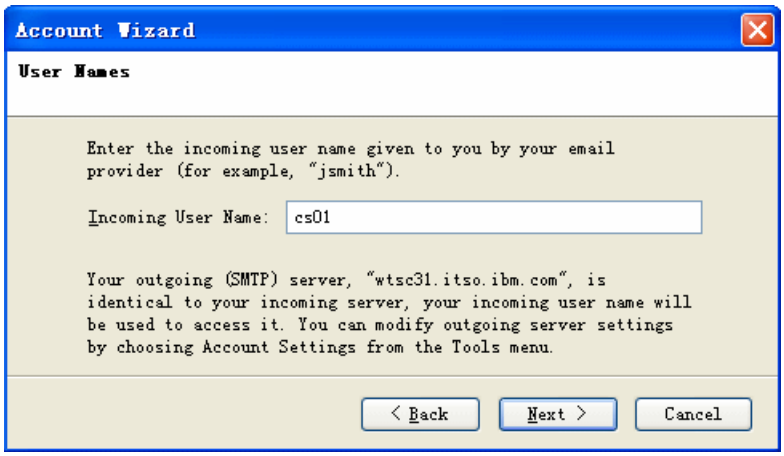


Figure 7-13 Fourth panel of the Mozilla Thunderbird account wizard

The fifth panel of the wizard asks for a name for the account. We used the name shown in Figure 7-14.

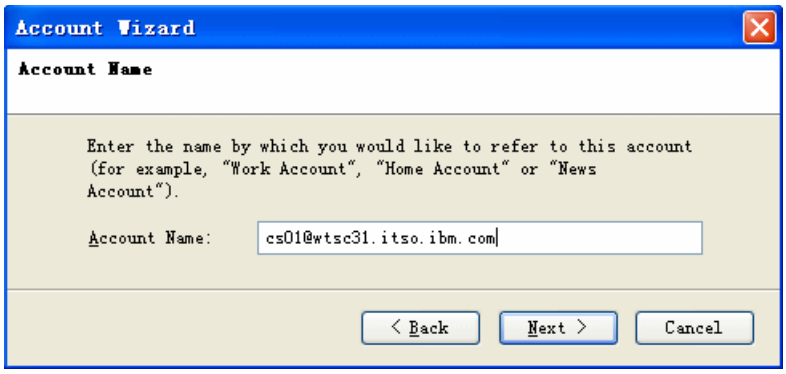


Figure 7-14 Fifth panel of the Mozilla Thunderbird account wizard

The final panel in the account wizard asks for you to confirm the settings entered in the previous panels. Our confirmation panel is shown in Figure 7-15.

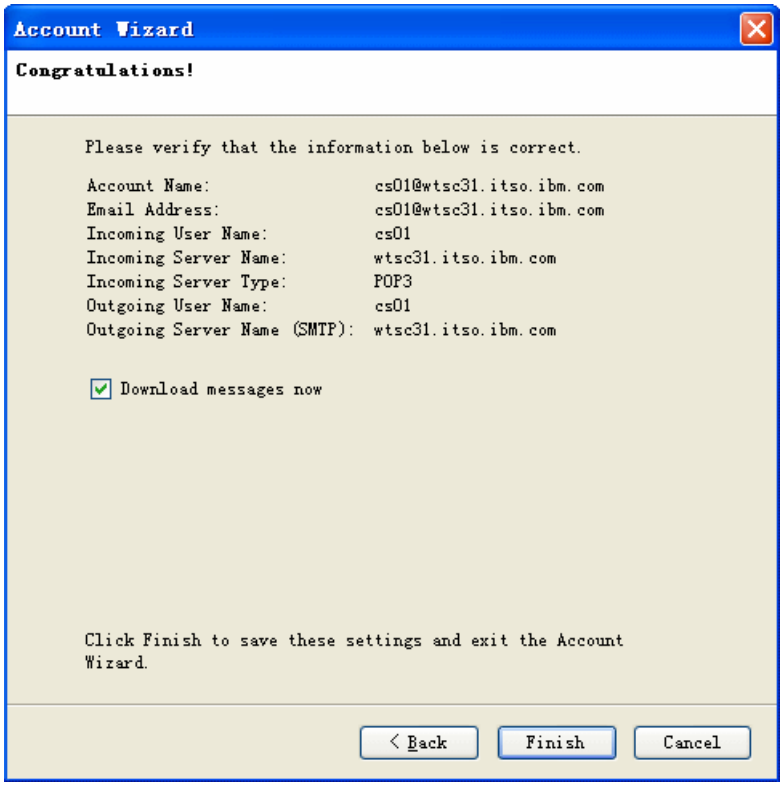


Figure 7-15 Final panel of the Mozilla Thunderbird account wizard

Now that Mozilla Thunderbird was configured to communicate with our system running sendmail, we sent a test message to ourself to test the environment. Figure 7-16 on page 308 shows the message that we sent.

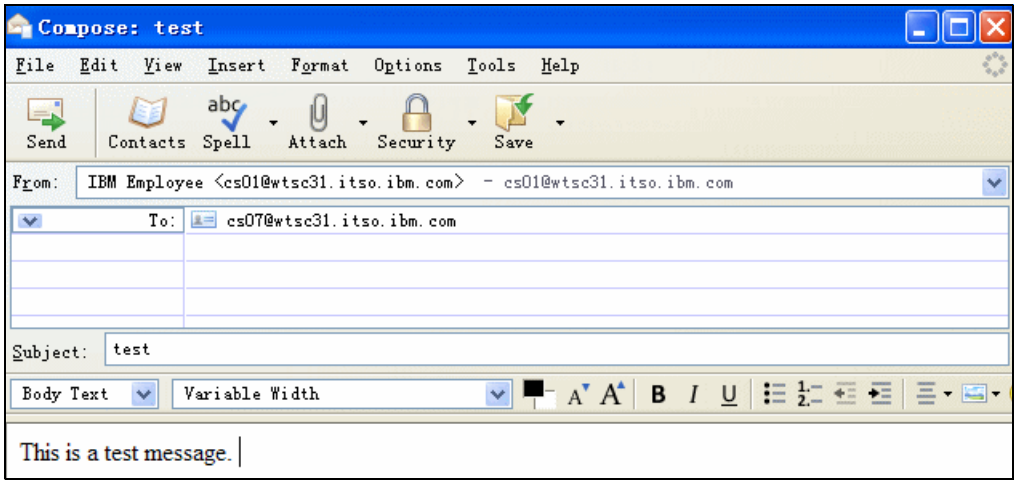


Figure 7-16 Outgoing test message

After a brief wait of approximately 10 seconds, Thunderbird confirmed that the email message had been sent. At this point we have shown that sendmail is working properly and queuing up messages for delivery. To confirm that the message was delivered properly and to test popper's ability to transfer the message to our Mozilla Thunderbird mail user agent, we need to check for new mail.

After a brief wait, we clicked the **Get Mail** button in Thunderbird and received the message shown in Figure 7-17.

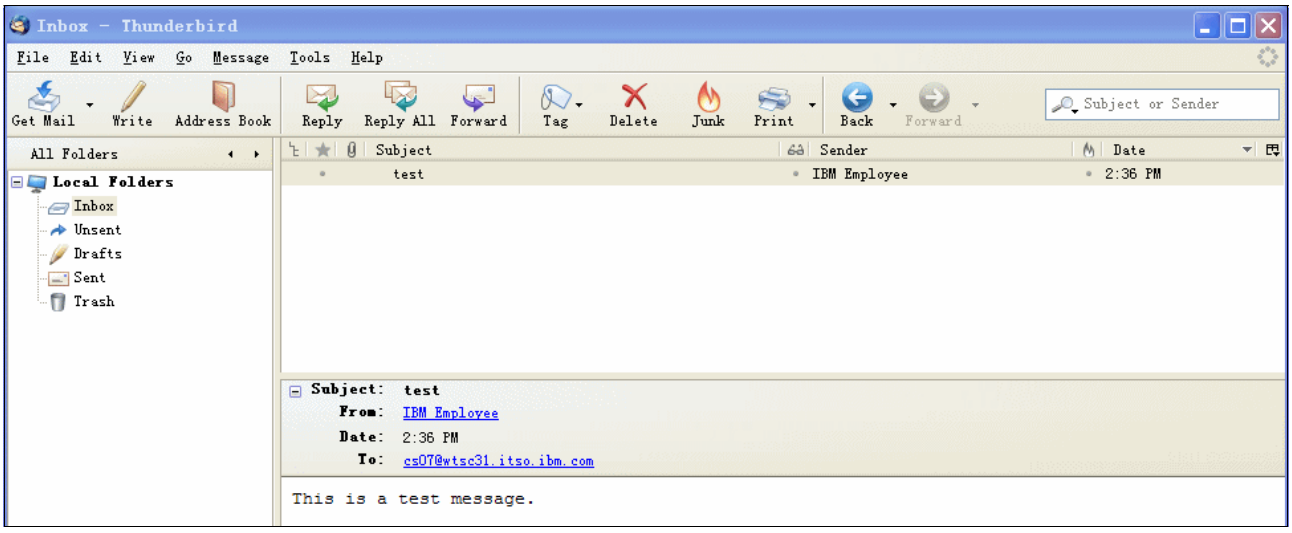


Figure 7-17 Inbound message

We have now shown that both sendmail and popper are up and running and able to deliver mail locally.

7.5 Using sendmail as a client

In this section we discuss the use of sendmail as an email client. The topics are:

- Description of the sendmail client
- Configuration tasks for the sendmail client
- Verification of the sendmail client

7.5.1 Description of the sendmail client

Sendmail is also capable of acting as a client that can be used to send email. Sendmail supports MIME attachments and also has built-in security features such as TLS/SSL sockets.

Sendmail can be complex. Sendmail is incapable of interfacing with JES. If you want to submit batch job email, use the SMTP batch client rather than the sendmail client. See Figure 7-5 on page 283 to review the role of an SMTP client. Remember that a client can be an individual user or an SMTP command.

7.5.2 Configuration tasks for the sendmail client

We compose a message on z/OS and use sendmail to send it to another email address. In this section, we also discuss how to create an email attachment in our message. We attach a GIF image of the IBM logo to the bottom of our message.

The logo we use as an attachment is shown in Figure 7-18. We have sent image file with FTP to the z/OS UNIX file system and placed it in `/tmp/ibm-logo.gif`.

Figure 7-18 GIF image of the IBM logo in /tmp/ibm-logo.gif

The configuration tasks for the sendmail client are:

- Set up sendmail and popper clients.
- Compose an RFC 822-compliant message.
- Encode the attachment.
- Join the encoded file with the text message.
- Add MIME attachment headers to the message.
- Invoke sendmail in the client mode to send the message.

Set up sendmail and popper clients

First, set up sendmail and popper as described in 7.4.3, “Verification of sendmail and popper setup” on page 304. The configuration file, alias file, and local-host-names files are all required for sendmail, regardless of whether sendmail is run in server mode or client mode.

Compose an RFC 822-compliant message

Compose an RFC 822-compliant message in a file in the z/OS UNIX file system. RFC 822 specifies the format of a standard Internet email message. You can view RFC 822 on the Internet at:

`http://www.ietf.org/rfc/rfc822.txt`

An RFC 822-compliant message is one that simply contains To:, From:, Subject:, and Date headers, followed by a blank line before the text of the message. We composed our RFC

822-compliant message in the z/OS UNIX file system in the /tmp/test-msg file. The contents of that file are shown in Example 7-16.

Example 7-16 RFC 822-compliant message in /tmp/test-msg

```
To: cs07@wtsc31.itso.ibm.com
From: cs01@wtsc31.itso.ibm.com
Date: Fri Sep 28 00:39:28 2007
Subject: test message
Hello,
   This is a test message using sendmail.
LGT
```

Encode the attachment

To send binary data through email, you must encode the binary data into an email safe format. Modern email applications use an encoding called *base64* to encode binary files. We found the easiest way to perform base64 encoding is to download base64 encoding/decoding software from the Internet. We used the b64 program that is available at:

http://base64.sourceforge.net

After downloading the source, we compiled the b64.c program from the z/OS UNIX shell using the **cc -o b64 b64.c** command.

Important: Before compiling the b64.c program, change `fprintf( outfile, \r\n )` to `fprintf( outfile, \n )` in the `b64.c` file. This way, b64 will not append the unnecessary ^M characters.

The ^M character represents an end-of-line character (carriage return) that is used in DOS-based systems, but is not used for end-of-line on UNIX-based systems. UNIX-based systems only use the line feed character to mark the end of a line.

Then we issued **b64 -e /tmp/ibm-logo.gif /tmp/ibm.gif** to encode the attachment. After encoding we used the command **cat /tmp/ibm.gif** to view the base64 encoded image. Example 7-17 shows the contents of our base64 encoded file.

Example 7-17 base64 encoded image in /tmp/ibm.gif

```
R0lGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR
zIuz3nzHtHyGVTOhYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

Join the encoded file with the text message

We use the z/OS UNIX **cat** command to join our text message with our 7-bit ASCII image file:

```
cat /tmp/test-msg /tmp/ibm.gif > /tmp/test-message
```

The contents of the file /tmp/test-message are shown in Example 7-18.

Example 7-18 /tmp/test-message after joining the text and binary files

```
To: cs07@wtsc31.itso.ibm.com
From: cs01@wtsc31.itso.ibm.com
Date: Fri Sep 28 00:39:28 2007
Subject: test message
```

```
Hello,
   This is a test message using sendmail.
LGT
ROlGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR
zIuz3nzHtHyGVTOhYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

Add MIME attachment headers to the message

Multipurpose Internet Mail Extensions (MIME) are additional headers that can be included in email messages. In email messages, an attachment is simply binary data surrounded by MIME headers. A email client capable of handling attachments simply looks for the MIME headers and separates the email text from the binary attachment.

The headers required for our MIME message include the MIME-Version header and Content-Type header at the top of the message, with a subtype of multipart/mixed that identifies a unique boundary string for each part of the message. In each separate part of the message we added Content-Type and Content-Transfer-Encoding headers.

The headers we added to our message are shown in Example 7-19.

RFC 2045 and RFC 2046 describe MIME in full detail, and are available at:

- http://www.ietf.org/rfc/rfc2045.txt
- http://www.ietf.org/rfc/rfc2046.txt

Example 7-19 /tmp/test-message with MIME headers

```
To: cs07@wtsc31.itso.ibm.com
From: cs01@wtsc31.itso.ibm.com
Date: Fri Sep 28 00:39:28 2007
Subject: test message
MIME-Version: 1.0
Content-Type: multipart/mixed;
 boundary="M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y"
This is a multi-part message in MIME format.
--M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y
Content-Type: text/plain
Content-Transfer-Encoding: 7bit

Hello,
   This is a test message using sendmail.
LGT

MWP
--M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y
Content-Type: image/gif;
 name="ibm.gif"
Content-Transfer-Encoding: base64
Content-Disposition: inline;
 filename="ibm.gif"

ROlGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR
zIuz3nzHtHyGVTOhYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
-M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y--
```

Invoke sendmail in the client mode to send the message

At this point, we were able to send our message; we invoked sendmail with the following command:

```
/usr/sbin/sendmail -C /etc/mail/sendmail.cf -bm -t < /tmp/test-message
```

7.5.3 Verification of the sendmail client

To verify that our attachment was successfully sent, we simply checked our email with Mozilla Thunderbird, which was set up as described in 7.4.3, “Verification of sendmail and popper setup” on page 304. After clicking **Get Mail** in Mozilla Thunderbird, we received the message shown in Figure 7-19.

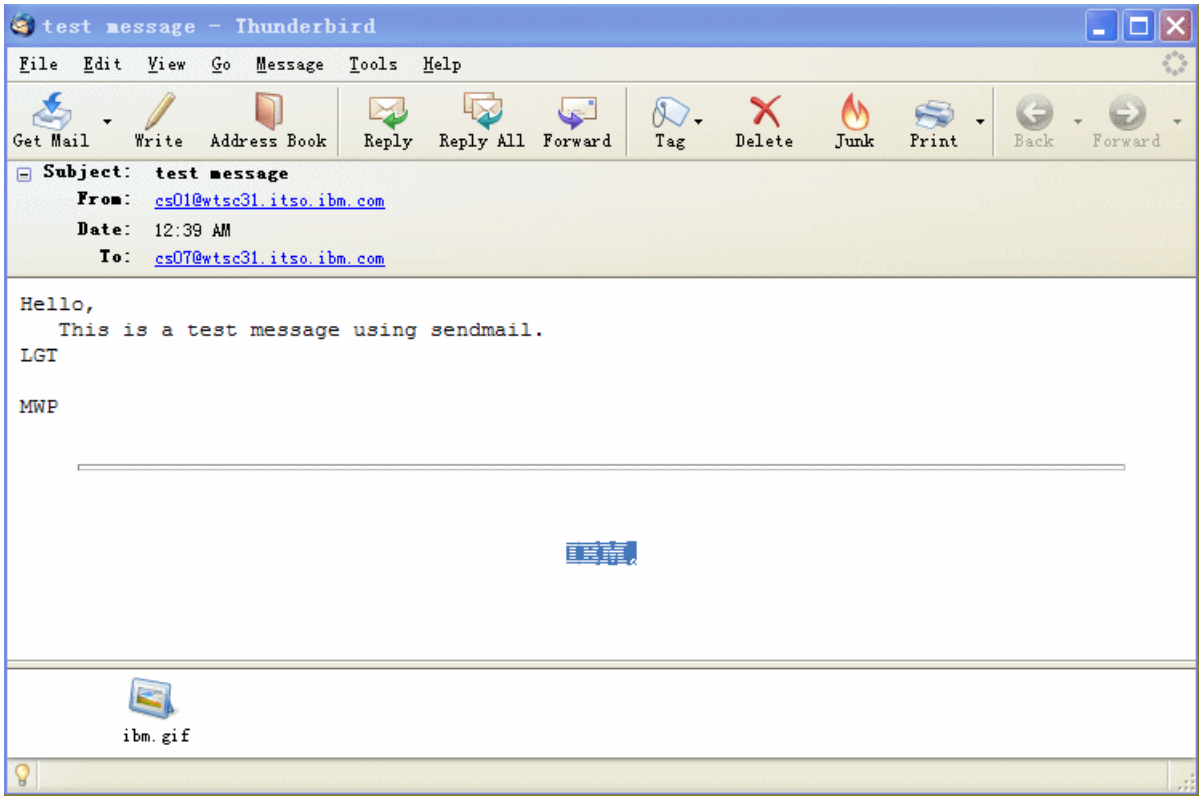


Figure 7-19 Message with attachment that was received from our z/OS system

7.6 Problem determination for the mail facilities

This section includes the following problem determination information:

- Problem determination tasks for the z/OS SMTP server
- Problem determination for sendmail and popper
- Problem determination for the sendmail client

7.6.1 Problem determination tasks for the z/OS SMTP server

If changes to the domain name server require you to resolve already queued mail again, use the SMSG SMTP EXPIRE command, or the MODIFY SMSG,EXPIRE command. See the SMSGAUTHLIST statement for the SMTP server configuration data set for its usefulness in

problem determination. You can also query operating statistics, such as mail delivery queues of the SMTP server. Several of the commands are shown in Table 7-1.

Table 7-1 Useful diagnostic commands for SMTP

TSO SMSG format	MVS MODIFY format
SMSG smtpproc HELP	F smtpproc,SMSG,HELP
SMSG smtpproc DEBUG or NODEBUG	F smtpproc,SMSG,DEBUG or NODEBUG
SMSG smtpproc TRACE or NOTRACE	F smtpproc,SMSG,TRACE or NOTRACE
SMSG smtpproc STARTEXIT or STOPEXIT	F smtpproc,SMSG,STARTEXIT or STOPEXIT
SMSG smtpproc STATS	F smtpproc,SMSG,STATS
SMSG smtpproc EXPIRE ipaddr	F smtpproc,SMSG,EXPIRE,ipaddr
SMSG smtpproc NUMQUEUE	F smtpproc,SMSG,NUMQUEUE
SMSG smtpproc QUEUES	F smtpproc,SMSG,QUEUES
SMSG smtpproc SHUTDOWN	F smtpproc,SHUTDOWN

These administrative tasks are discussed in more detail in:

- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781

7.6.2 Problem determination for sendmail and popper

The sendmail "bat" book documents the detailed internal traces available in sendmail. See the *sendmail, 4th Edition* by Costales, Assmann, Jansen, Shapiro, from O'Reilly Media, Inc. (ISBN 10: 0-596-51029-2).

Popper accepts a command-line parameter **`-t`**, which specifies the location of a trace file for detailed tracing. Popper also logs messages to the *mail* syslogd facility.

Following are hints and tips for configuration checking when you meet problems on starting and using sendmail:

- 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.
- A program-controlled environment is necessary for sendmail to run in daemon mode when BPX.DAEMON is enabled, because many functions of sendmail (especially daemon functions) require it to change the user ID (UID) without prompting for a password.
- The daemon must be started by root, as usual. Table 7-2 on page 314 shows the recommended security file permissions of files that sendmail might use.

Table 7-2 Sendmail permission table

Path	Type	Owner	Mode	Required or configurable
/	Directory	*root*	555 dr-xr-xr-x	Required
/usr	Directory	*root*	555 dr-xr-xr-x	Required
/usr/sbin	Directory	*root*	555 dr-xr-xr-x	Required
/usr/sbin/sendmail	File	*root*	755 -rwxr-xr-x	Required
/bin/sendmail	File	*smmsp*	755 -rwxr-xr-x	Configurable[a]
/etc/mail	Directory	*root*	555 dr-xr-xr-x	Configurable
/etc/mail/sendmail.cf	File	*root*	444 -r--r--r--	Configurable
/etc/mail/submit.cf	File	*root*	444 -r--r--r--	Configurable
/var/spool/mqueue	File	*sendmail*	700 -rwx------	Configurable
/var/spool/clientmqueue	File	*smmsp*	770 -rwxrwx---	Configurable

a. Used only with RACF program control systems

Note: When sendmail is attempting to canonify a host name, some broken name servers will return SERVFAIL (a temporary failure) on T_AAAA (IPv6) lookups. To allow sendmail to accept this behavior, ResolverOptions in the configuration file is set to WorkAroundBrokenAAAA by default.

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.

7.6.3 Problem determination for the sendmail client

The sendmail “bat” book documents the detailed internal traces available in sendmail. See *sendmail, 4th Edition* by Costales, Assmann, Jansen, Shapiro, from O'Reilly Media, Inc. (ISBN 10: 0-596-51029-2).

7.7 Additional information sources for mail servers

See the following sources for additional information:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User’s Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781
- *TCP/IP Tutorial and Technical Overview*, GG24-3376

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

8

z/OS UNIX Telnet server

The z/OS UNIX Telnet server, also known as otelnetd, enables remote Telnet clients to directly log on to the z/OS UNIX System Services shell without having to log onto TSO. The facility provides an ASCII line mode terminal screen interface instead of the 3270 full-screen interface offered by TSO. This chapter focuses on the z/OS UNIX Telnet server functions that are available in the z/OS Communications Server.

This chapter uses the terms *otelnetd* and *z/OS UNIX Telnet server* to refer to the Telnet server that provides access to the z/OS UNIX System Services shell. Other manuals might also refer to the same application as *oTelnet*, or *Telnet daemon*.

This chapter discusses the following z/OS UNIX Telnet server topics.

Section	Topic
8.1, “Conceptual overview of otelnetd” on page 316	The basic concepts of the z/OS UNIX Telnet server.
8.2, “z/OS UNIX Telnet server implementation” on page 318	Key characteristics of the z/OS UNIX Telnet server and why it might be important in your environment.
8.3, “Problem determination for otelnetd” on page 324	Commonly implemented z/OS UNIX Telnet server design scenarios, their dependencies, advantages, considerations, and our recommendations.
8.4, “Additional information sources for otelnetd” on page 324	Selected implementation scenarios, tasks, configuration examples, and problem determination suggestions.

8.1 Conceptual overview of otelnetd

As illustrated in Figure 8-1, otelnetd, the z/OS UNIX Telnet server, is one of the standard applications provided with the z/OS Communications Server. It interfaces with the z/OS UNIX Sockets through C-Sockets, passing its packets in and out of the Logical and Physical File Systems.

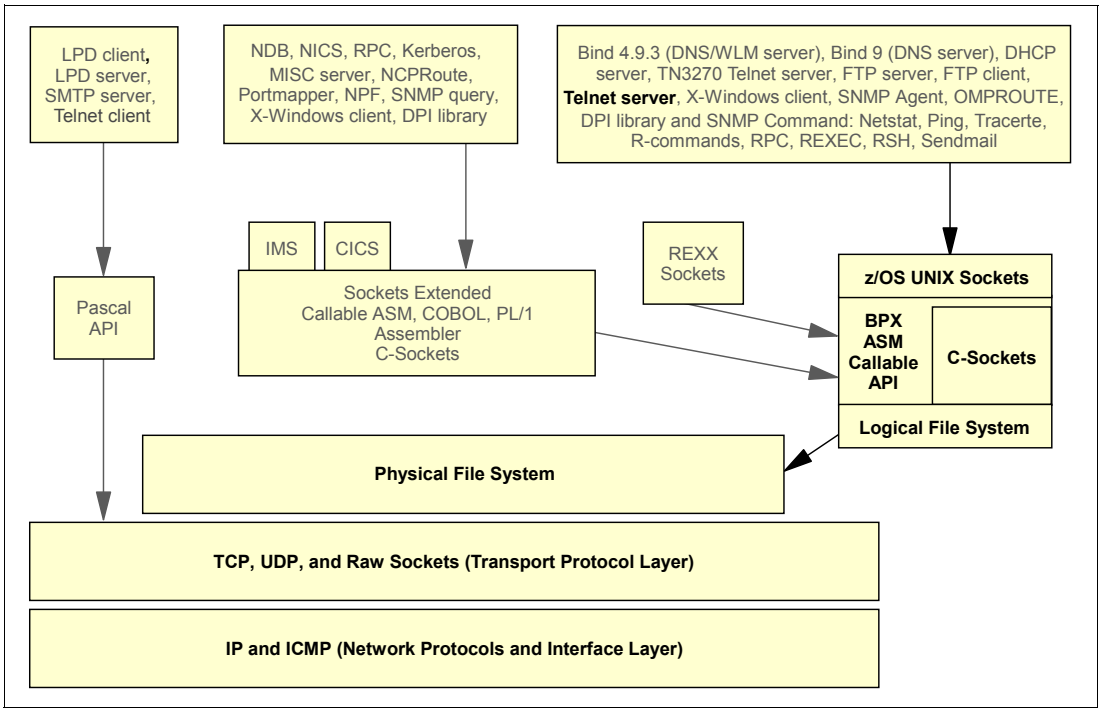


Figure 8-1 otelnetd application services

8.1.1 What is otelnetd

The z/OS UNIX Telnet server provides Telnet access directly to the z/OS UNIX shell, without requiring a TSO session. The z/OS Telnet server also provides different terminal capabilities. These features are useful if you have UNIX applications that run under z/OS UNIX.

8.1.2 How does otelnetd work

The z/OS UNIX Telnet server works in either character mode or line mode, but does not support SNA 3270 emulation (for 3270 support. See Chapter 2, “TN3270E Telnet server” on page 35). The z/OS UNIX server can optionally use Kerberos 5 authentication and DES encryption.

The z/OS UNIX Telnet server authenticates users, negotiates Telnet options with the clients, and then spawns a z/OS UNIX shell where z/OS UNIX commands can be executed. This process is illustrated in Figure 8-2.

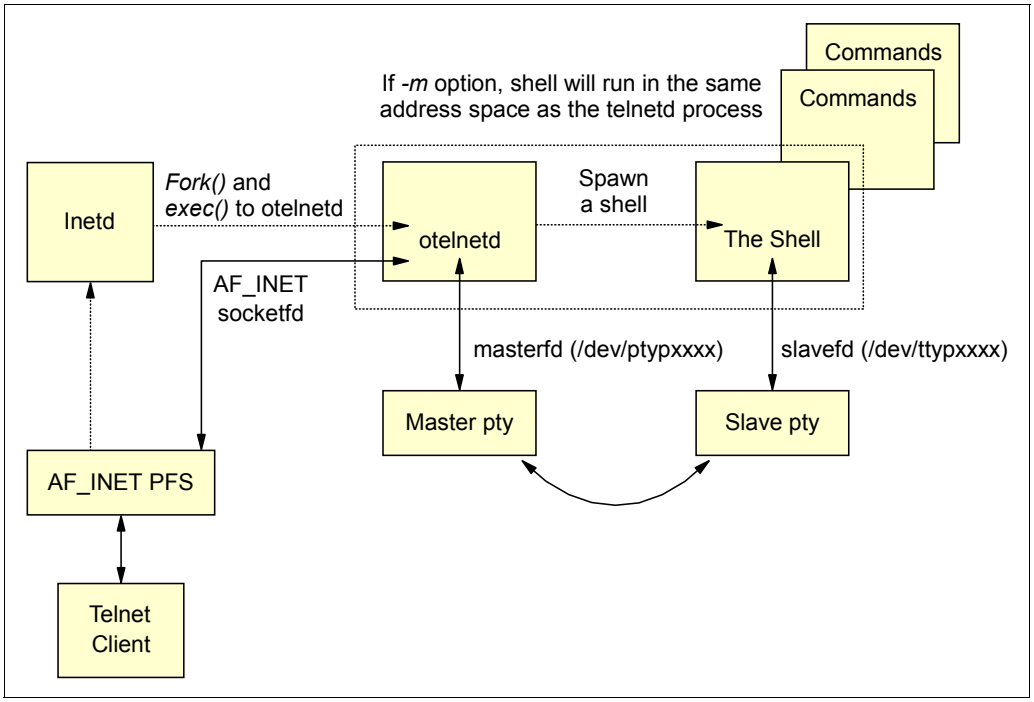


Figure 8-2 otelnetd interactions with INETD and z/OS UNIX

8.1.3 How can otelnetd be applied

The z/OS UNIX Telnet server is a simple application with limited configuration options. We recommend running the z/OS UNIX Telnet server only if you require direct access to the z/OS UNIX shell. The z/OS UNIX Telnet server is only needed if you intend to access the z/OS UNIX shell without first accessing TSO or if you intend to use applications that depend on the special terminal capabilities available in z/OS UNIX.

The z/OS UNIX Telnet server is started by INETD, which is discussed in Chapter 6, “INETD” on page 263. After INETD has created a TCP connection with a client, INETD forks and executes an instance of otelnetd.

8.2 z/OS UNIX Telnet server implementation

The relationships between otelnetd, the TN3270 Telnet server, and INETD are shown in Figure 8-3.

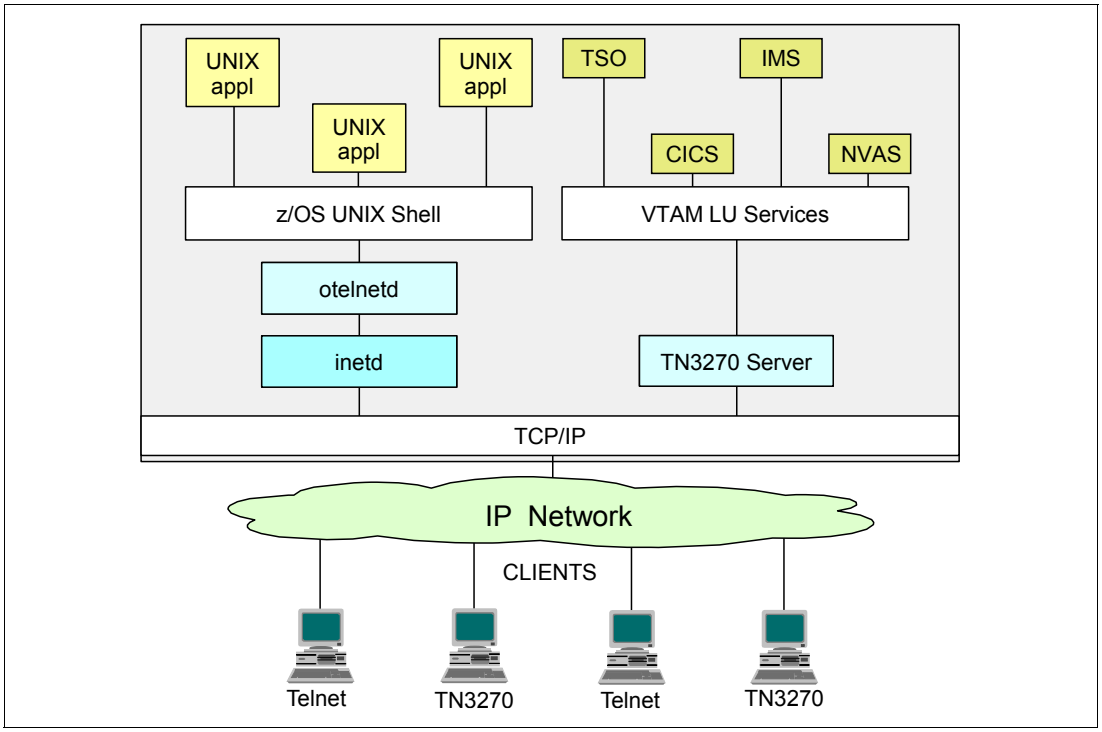


Figure 8-3 INETD and otelnet relationship

This section includes the following topics:

- Description of the otelnetd server
- Configuration tasks for otelnetd
- Activation and verification of otelnetd

8.2.1 Description of the otelnetd server

This section discusses otelnetd in a typical environment, with no authentication or encryption. The z/OS UNIX Telnet server is easy to implement and provides a quick method to access the z/OS UNIX shell. The z/OS Telnet server requires an active TCP/IP stack and INETD. We also highly recommend that you start syslogd to manage messages that can be generated by the otelnetd server.

Note: The configuration we show can send sensitive data over insecure communication channels. If authentication and security is vital to your environment, then Kerberos security or AT-TLS provide options to secure the z/OS UNIX Telnet session. Kerberos is discussed in the *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926. AT-TLS is discussed in *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

8.2.2 Configuration tasks for otelnetd

To start the z/OS UNIX Telnet server, INETD must be active and contain a line in its configuration file for otelnetd. A port must also be reserved in `/etc/services` with the same service name.

The configuration tasks are as follows:

- Plan the otelnetd environment
- Reserve a PORT in PROFILE.TCPIP
- Create a port entry in /etc/services
- Update the INETD configuration file for otelnetd

Note: If you have already set up and started INETD per our recommendations in Chapter 6, “INETD” on page 263, then you have already completed the necessary configuration and otelnetd might already be active.

Plan the otelnetd environment

Because z/OS is a native EBCDIC system, and the Telnet protocol is a native ASCII protocol, a method to perform ASCII/EBCDIC translation is required. The ASCII/EBCDIC translation is performed by otelnetd and relies on the **chcp** shell command to provide code page conversions. The **chcp** command supports both single-byte character set (SBCS) and double-byte character set (DBCS) code pages.

When a **chcp** shell command is executed, the **otelnetd** process is informed of the code page change through urgent data over the pseudo terminal connection (the master/subordinate pty interface). If a user selects to change the code page, the **chcp** command can be executed as part of the user’s login process, for example, using the user’s `$HOME/.profile` or `$HOME/.setup` shell scripts.

For more information about the **chcp** command, see *z/OS UNIX System Services Command Reference*, SA22-7802.

Consider the following topics before implementing **otelnetd**:

- Banner pages
- Pseudoterminals
- The terminal capabilities database
- Kerberos

Banner pages

You can create two environment-specific banner pages:

- Pre-login
- Post-login

You can define the pre-login banner in the `/etc/otelnetd.banner` file (shown in Example 8-1). It displays at the client workstation before the user ID and password are entered.

Example 8-1 The /etc/otelnetd.banner file for the pre-login banner

```
**********************************************************************
*                                                                    *
*  Welcome to z/OS UNIX System Services                              *
*                                                                    *
* This is /etc/otelnetd.banner the pre-login banner for Telnet Server *
*                                                                    *
**********************************************************************
```

```
EZYTE27I login: cs03
EZYTE28I cs03 Password:
```

You can define the post-login banner in the /etc/banner file (shown in Example 8-2). It displays at the client workstation after the user ID and password are entered.

Example 8-2 The /etc/banner file for the post-login banner

```
**********************************************************************
*                                                                    *
*               Welcome to z/OS UNIX System Services                 *
*                                                                    *
* This is /etc/banner (the post-login banner for the Telnet Server)  *
*                                                                    *
**********************************************************************

IBM
Licensed Material - Property of IBM
5694-A01 Copyright IBM Corp. 1993, 2009
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government Users Restricted Rights -
Use,duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

CS03 @ SC30:/u/cs03>
```

Pseudoterminals

Pseudoterminal files are created in the z/OS UNIX file system, in the /dev directory. The file names are ptyp*nnnn* and ttyp*nnnn*, where *nnnn* is a number from 0000 to 9999. These files are used in pairs by the otelnetd server. z/OS UNIX creates pseudoterminal pairs dynamically as needed. In the rare case where the number of pseudoterminals needs to be increased, you can manually issue the **mknod** command to create additional pseudoterminals. The maximum number of pseudoterminals is constrained by the MAXPTYS statement in your BPXPRMxx parmlib member. For more information regarding pseudoterminals and the MAXPTYS parameter see *z/OS UNIX System Services Planning*, GA22-7800.

The terminal capabilities database

For telnet sessions, you might need some definitions of terminal capabilities to let z/OS UNIX application programs manipulate the terminal output correctly. These definitions are kept in the terminfo database.

A UNIX system, like other operating systems, must have at least one terminal attached to it. In the early days these were typewriters having keyboard input and a paper output. These terminals were later replaced by video display units (VDUs), which behave like the typewriter did earlier—the paper is just replaced by the screen. These VDUs understand a data stream that contains ASCII characters to be printed, including control characters such as carriage return or new line. Differences appeared when individual manufacturers started to add

specific commands to their terminals. Specific commands, for example, can include cursor movements (up, down, left, right, and so forth).

When one uses simple printing commands, such as **cat**, there is no obvious difference among terminals. However, these differences must be taken into account as soon as one runs a program that uses special commands. This is mostly the case with editors, such as vi or emacs. These programs need to know, for example, how to move the cursor to a specific location.

z/OS UNIX and other UNIX systems that have their roots in System V UNIX create a database named terminfo that contains definitions of all of the capabilities of each terminal. The terminfo database is shipped as part of z/OS. The database is populated with the terminal types defined by ibm.ti, dec.ti, wyse.ti, ansi.ti, and dtterm.ti. The database is in the directory /usr/share/lib/terminfo and the source files are in /samples. Each type of terminal that is defined has a corresponding file with the `.ti` suffix. If you need to recreate the terminfo database, run the **tic** utility. For example, to define an IBM terminal for the terminfo database, specify from the shell environment:

```
tic /samples/ibm.ti
```

To define terminal types such as VT100 and VT220, specify from the shell environment:

```
tic /samples/dec.ti
```

Table 8-1 shows the directories and files shipped as part of z/OS UNIX or created by the **tic** command.

Table 8-1 Terminfo directory structure and contents

Directory	Terminal definition files
a	aixterm, aixterm-m, aixterm-m-old, aixterm-old
c	cdef
d	dtterm, dumb
h	hft, hft-c-old, hft-m-old, hft-nam-old, hft-c, hft-m, hft-nam, hft-old
i	ibmpc, ibmpcc, ibmpdp, ibm3101, ibm3151, ibm3151-noc, ibm3151-S, ibm3151-132, ibm3151-25, ibm3151-51, ibm3151-61, ibm3152, ibm3152-PS2, ibm3152-132, ibm3152-25, ibm3161, ibm3161-C, ibm3162, ibm3162-132, ibm3163, ibm3164, ibm5081, ibm5081-old, ibm5081-113, ibm5081-56, ibm5151, ibm5154, ibm5550, ibm5570, ibm6153, ibm6153-40, ibm6153-90, ibm6154, ibm6154-40, ibm6154-90, ibm6155, ibm6155-113, ibm6155-56, ibm8503, ibm8507, ibm8512, ibm8513, ibm8514, ibm8515, ibm8604
j	jaixterm, jaixterm-m
l	lft, lft-pc850
L	LFT, LFT-PC850
v	vs100, vs100s, vt100, vt100-am, vt100-nam, vt100x, vt200, vt200-8, vt320, vt320, vt320-8, vt330,vt330-8, vt340
w	wyse100, wyse30, wyse350, wyse50, wyse50-2, wyse60, wyse60-AT, wyse60-PC, wyse60-316X, wy100, wy30, wy350, wy50, wy50-2, wy60, wy60-AT, wy60-PC, wy60-316X
x	xterm, xterms

Whenever a shell environment is created, the terminal type of the client user is registered in the environment variable TERM. The TSO OMVS command environment handles TERM as though it were set to TERM=dumb. In a telnet session the TERM environment variable is set to the terminal type that the telnet client uses or emulates.

To use terminal types that are not supported by CS for z/OS IP you can create a directory to store local terminal definitions and use the TERMINFO environment variable to define the location of that directory. This environment variable was implemented to allow typical UNIX clients with GUI windowed command-line prompts access to the z/OS UNIX Telnet server. This variable should be used if there are installation defined terminfo definitions. The environment variable can be passed directly to otelnetd with the -T command-line option, and should be set in /etc/profile or in the user's login script. See the *z/OS UNIX System Services Planning*, GA22-7800, for more information about terminal definitions.

If you are interested in more information about termcap and terminfo, see the book *termcap & terminfo*, published by O'Reilly Media, Inc.

Kerberos

The z/OS UNIX Telnet server includes support for Kerberos 5 authentication and encryption over IPv4 connections. Kerberos is not supported on IPv6 connections. The Kerberos support is provided by z/OS Security Server. See *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926, for more information.

Reserve a PORT in PROFILE.TCPIP

Although not required, we recommend that you reserve the port to be used for otelnetd in PROFILE.TCPIP, using the INETD job name. If you do not reserve a port, it is possible that some other application will bind to the ports normally reserved for otelnetd. Our PORT statement for otelnetd is shown in Example 8-3 and reserves the ports for job name INETD1. We also chose to bind otelnetd to a specific DVIPA address so that otelnet could listen on port 23 on one address and TN3270 could listen on port 23 on a different IP address.

Example 8-3 PORT statement in PROFILE.TCPIP

```
PORT
  23 OMVS BIND 10.1.9.21 ; OE Telnet Server
```

Create a port entry in /etc/services

An entry in `/etc/services` is required to map the otelnet service name to a port number. Example 8-4 shows our entry in `/etc/services` and maps the service `otelnet` to port 23 tcp.

Example 8-4 Statement in /etc/services

```
otelnet         23/tcp
```

Update the INETD configuration file for otelnetd

An entry is required in the INETD configuration file so that INETD will set up a listening socket for otelnetd. The application otelnetd can accept a number of command-line parameters that are specified in the INETD configuration file. All of the possible command-line parameters are discussed in detail in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. We

use only the -m option, which has the same affect as the _BPX_SHAREAS environment variable, allowing forked or spawned processes to coexist in the same address space. Example 8-5 shows our INETD configuration file entry.

Example 8-5 Statement in INETD configuration file

```
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
```

8.2.3 Activation and verification of otelnetd

Both z/OS UNIX Telnet and TN3270 listen on port 23 by default. There are two methods to run both servers concurrently:

- Start TN3270 on port 23, and assign a different port to z/OS UNIX Telnet. Any clients connecting to the z/OS UNIX Telnet server will need to specify the alternate port.
- Start TN3270 on port 23 on one IP address, and start z/OS UNIX Telnet on port 23 but bound to a different IP address. Any clients wanting to connect to the z/OS UNIX Telnet server will need to specify that different IP address or host name for the z/OS UNIX Telnet server.

Verification of otelnetd is simple: use your favorite Telnet client, connect to the address and port that otelnetd is listening on, and log in. If you can log in and everything is readable, then otelnetd is working correctly.

Figure 8-4 shows that we successfully logged in to our environment.

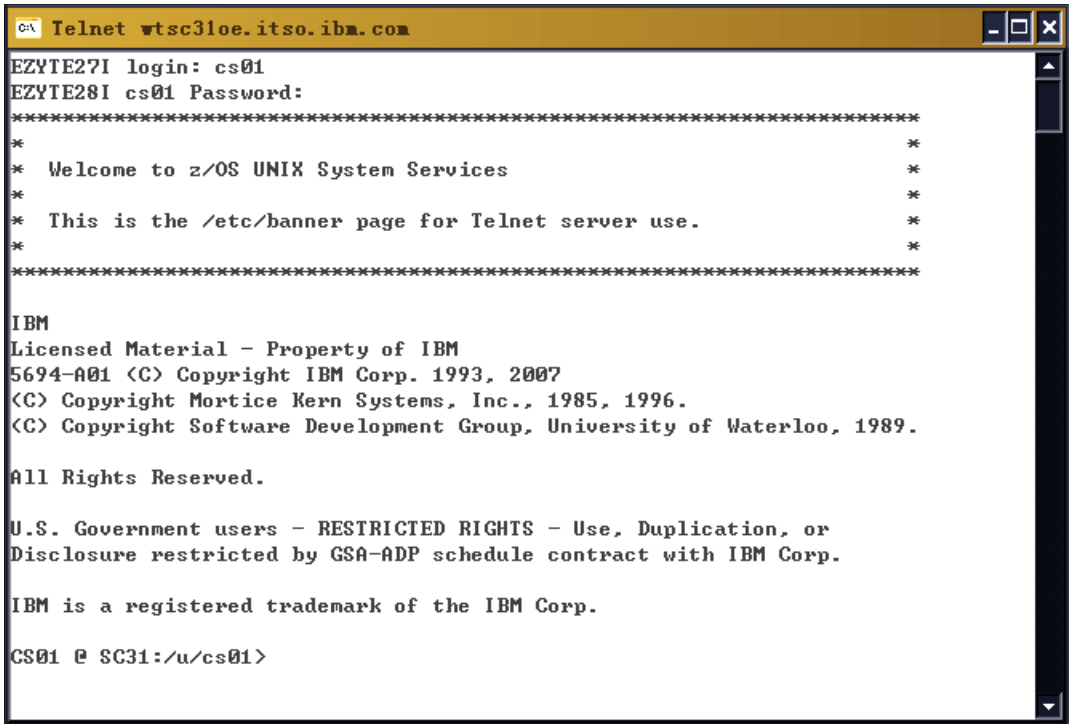


Figure 8-4 Successful login

Support for bypassing host name lookup in otelnetd

Typically, the routines that are used to obtain a host name based on its IP address are *gethostbyaddr* or *getnameinfo*. In some cases, the routines can have or can cause problems. For example, failures in the response of DNS can lead to a delay in processing. Or, if the IP address is not defined to the DNS, these routines will fail.

These routines are issued by otelnetd using the client address. If DNS does not have information for the user, the message EZYTE52E is issued by the z/OS UNIX Telnet server as shown in Example 8-6. These messages can potentially fill up syslogd log files and cause a delay in logging on.

Example 8-6 Error message EZYTE52E

```
EZYTE52E Couldn't resolve your address into a host name.
 IP address is 10.1.2.33
 EDC9501I The name does not resolve for the supplied parameters. rsn = 1
 EZYTO03E Incoming session is not from a registered host
```

The solution is to allow for the control of gethostbyaddr or getnameinfo lookup by otelnetd. The **-g** parameter disables the issuing of gethostbyaddr or getnameinfo using the client IP address to resolve the client host name. It is coded in the `/etc/inetd.conf` file, as follows:

```
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m -g
```

If the **-U** parameter is also specified, the **-g** parameter is ignored and message EZYTE90W is issued to syslogd, warning that the **-g** parameter has been overridden by the **-U** parameter. The **-U** parameter causes otelnetd to drop connections from any IP address that cannot be mapped back into a symbolic name by the gethostbyaddr or the getnameinfo routines.

```
EZYTE9OW Parameter -g ignored. Parameter not valid when -U coded
```

Note: The WHO command does not have the client host name because it is no longer learned by the gethostbyaddr or the getnameinfo routines.

8.3 Problem determination for otelnetd

The z/OS UNIX Telnet server includes a detailed trace, enabled with the **-t** command-line parameter, and can display information about the Telnet session with the **-D** command line option. *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, provides a detailed example of the trace output generated by otelnetd.

8.4 Additional information sources for otelnetd

See the following sources for additional information about otelnetd:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

9

Remote execution

Remote execution servers enable execution of commands that have been received from a remote client. These servers run the remote execution Command Daemon (REXECD), which supports both the Remote Execution Protocol (REXEC) and the Remote Shell (RSH) protocol. This chapter focuses on the remote execution functions that are available in the z/OS Communications Server.

This chapter discusses the following topics.

Section	Topic
9.1, “Conceptual overview of remote execution” on page 326	The basic concepts of remote execution and remote shell.
9.2, “TSO remote execution server” on page 331	How to set up and use a remote execution server in the TSO environment.
9.3, “z/OS UNIX remote execution server” on page 338	How to set up and use a remote execution server in the UNIX environment.
9.4, “REXEC TSO client command using user ID/password” on page 342	How to use the TSO client REXEC command with a user ID and password on the command.
9.5, “REXEC TSO client command using the NETRC data set” on page 347	How to use the TSO client REXEC command when using the NETRC data set to provide the user ID and password.
9.6, “REXEC UNIX client command” on page 353	How to use the UNIX client **rexec** command.
9.7, 9.7, “Problem determination for z/OS remote execution facilities” on page 355	Improvements in JES spool when REXECD is started with PURGE = N and others situation.
9.7, “Problem determination for z/OS remote execution facilities” on page 355	Problem determination techniques for the TSO and UNIX environments for remote execution servers and clients.
9.8, “Additional information sources for remote execution and remote shell” on page 360	References to additional reading sources for remote execution.

9.1 Conceptual overview of remote execution

REXEC and RSH are standard applications that are provided with the z/OS Communications Server to support remote execution services, as illustrated in Figure 9-1. The remote execution utilities use z/OS UNIX sockets to pass data through the Logical and Physical File Systems.

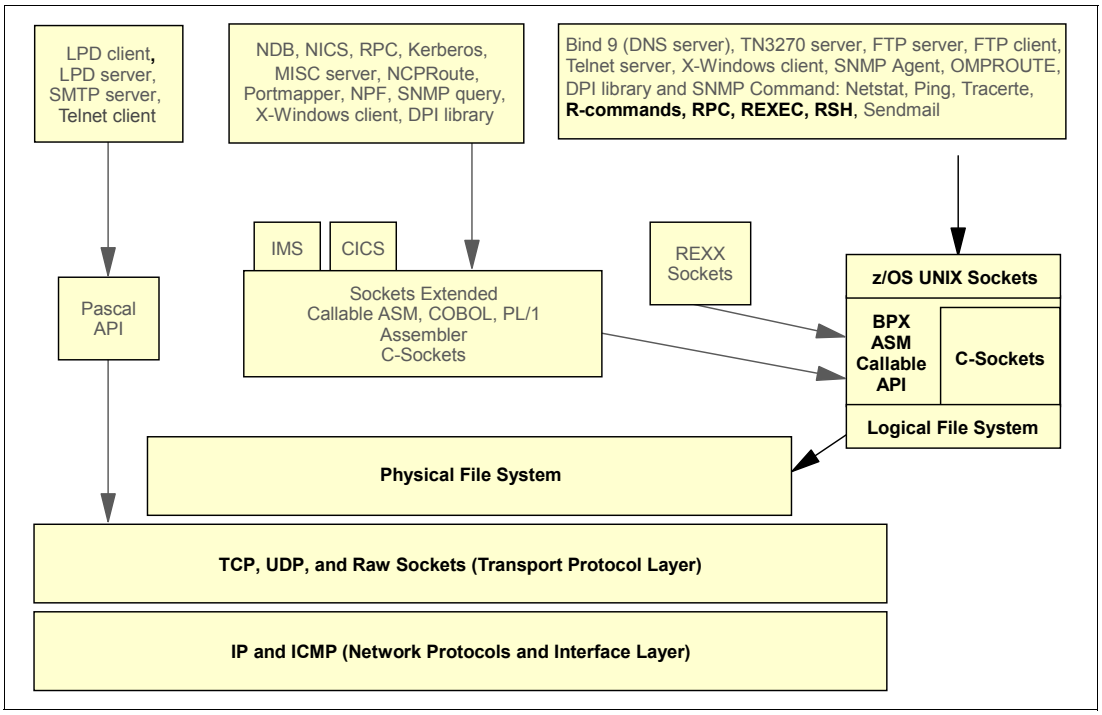


Figure 9-1 z/OS remote execution services

This section discusses the following topics:

- What is remote execution
- How does remote execution work
- How can remote execution be applied

9.1.1 What is remote execution

A client system might need to execute commands on another system without logging onto that target system directly. Remote execution is a simple solution for that requirement. Processes on one system in the IP network might need to initiate or post processes on another system to keep their environments synchronized. These processes might be automated, event driven, or manually activated by an operator. A system cannot have the software necessary to communicate with another system. An operator cannot have a terminal or terminal emulation software that is compatible with the other platform to log on directly to that target system. Remote execution is a command-level capability rather than an application-level one. Rather than enabling application-to-application communication, it establishes a low-level command submission and response retrieval process between two disparate systems that might otherwise not be able to communicate.

The z/OS Communications Server includes the following remote execution functionality:

- A TSO remote execution server (REXECD) and client (REXEC)
- A z/OS UNIX remote execution server (orexecd) and client (orexec/rexec)
- A z/OS UNIX remote shell server (orshd) and client (orsh/rsh)
- A TSO remote shell client

Note: The TSO remote execution server (REXECD) also supports the RSH protocol. Therefore a separate TSO remote shell server is not included in the z/OS Communications Server.

Terminology

Terminology and acronyms can be confusing. The documents referenced in 9.8, “Additional information sources for remote execution and remote shell” on page 360, are not necessarily consistent with their names and terminology when discussing the remote execution environments. We have consolidated the various names and terms associated with these two servers into a concise list to help you understand the references when you read them:

- The three terms *remote execution server*, *TSO remote execution server*, and *TSO RXSERVE daemon* refer to the single server *REXECD*, which is executed in the MVS environment as a started task.
  - REXECD is the distinguishing name for this TSO remote execution server.
  - SYS1.SEZAINST(RXPROC) is the sample JCL member for REXECD.
  - RXSERVE is the system proclib member name given to REXECD.
  - RXSERVE is the started task name used in the PROFILE.TCPIP data set.
  - The TSO RXSERVE daemon is the RXSERVE started task.
  - The RXSERVE task accepts and processes both commands, REXEC and RSH.
- The three terms *UNIX remote execution server*, *z/OS UNIX remote execution server*, and *z/OS UNIX REXECD server* refer to the single server *orexecd*, which is initiated by INETD in the UNIX environment.
  - *orexecd* is the distinguishing name for this z/OS UNIX remote execution server.
  - *rexecd* is considered a synonym of orexecd throughout the references.
  - orexecd resides in `/usr/sbin/orexecd`.
  - The orexecd server accepts and processes the **rexec**/**orexec** command.
- The three terms *Remote Shell Server*, *UNIX daemon*, and *z/OS UNIX rshd* refer to the same server *orshd*, which is initiated by INETD in the UNIX environment.
  - *orshd* is the distinguishing name for this z/OS UNIX remote shell server.
  - *remote shell*, *rsh*, and *rshd* are all used in the manuals to refer to rshd.
  - orshd resides in `/usr/sbin/orshd`.
  - The orshd server accepts and processes the **rsh**/**orsh** command.
- The term *remote execution protocol* generically refers to the support of the **rexec command**, whether discussing a server or client.
- The term *remote shell protocol* generically refers to the support of the **rsh command**, whether discussing a server or client.
- The client commands **REXEC** and **RSH** are used in the TSO environment, either in an interactive TSO session or in a batch TSO job.
- The client commands **rexec**, **orexec**, **rsh**, and **orsh** are used in the z/OS UNIX environment.
  - **rexec** and **orexec** are synonyms, and are used interchangeably.
  - **rsh** and **orsh** are synonyms, and are used interchangeably.

9.1.2 How does remote execution work

Figure 9-2 shows the relationship of the remote execution servers to the TCP/IP stack and the remote clients.

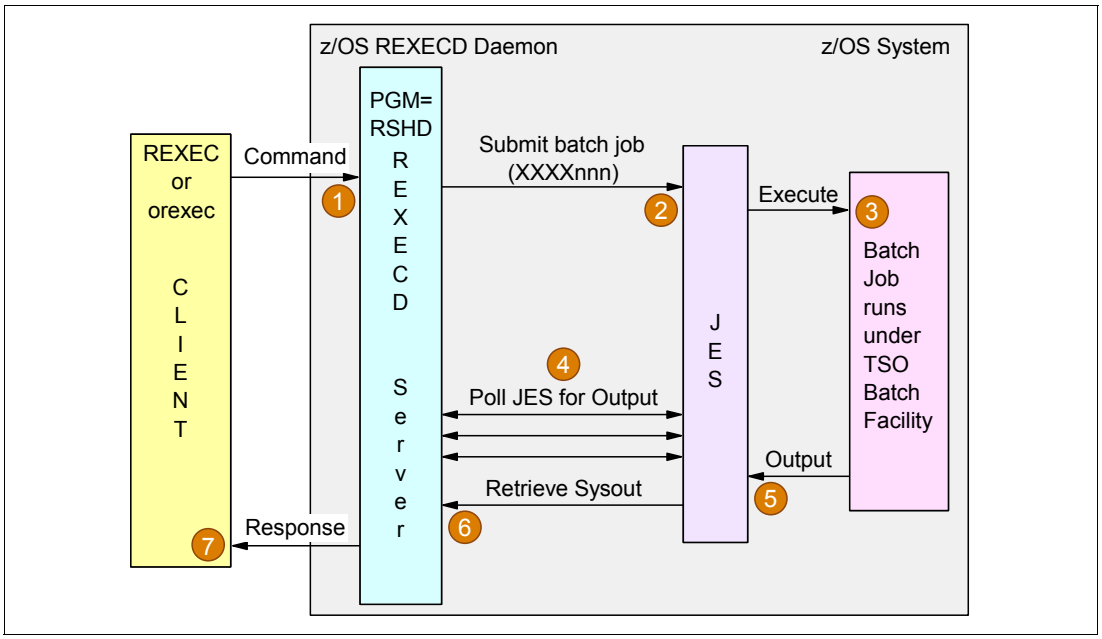


Figure 9-2 Remote execution client/server relationship

TSO remote execution server

The TSO remote execution *server* enables execution of TSO commands that have been received from a remote client. This server runs the remote execution Command Daemon (REXECD), which supports both the remote execution (REXEC) protocol and the Remote Shell (RSH) protocol.

TSO remote execution client

The TSO remote execution *client* is a TSO command that enables execution of a command on a remote system with the output returned to the TSO session. The TSO remote execution client can be run from TSO or in batch. When run from batch, the output of the remote command is returned to the batch job log. The remote host need not be a z/OS system, but can be any system with an REXEC server.

z/OS UNIX remote execution server

The z/OS UNIX remote execution server enables execution of z/OS UNIX commands that have been received from a remote client. Unlike the TSO remote execution server, this server runs just the remote execution Command Daemon (orexecd) supporting the remote execution (REXEC) protocol only.

z/OS UNIX remote execution client

The z/OS UNIX remote execution client, such as the TSO remote execution client, enables execution of a command on a remote system with the output returned to your z/OS UNIX System Services shell.

z/OS UNIX remote shell server

The z/OS UNIX remote shell server allows for execution of z/OS UNIX commands that have been received from a remote client. It is similar to the z/OS UNIX remote execution server, but uses a different protocol.

z/OS UNIX remote shell client

The z/OS UNIX remote shell client, like the TSO remote execution client, enables execution of a command on a remote system with the output returned to your z/OS UNIX System Services shell. The remote host need not be a z/OS system, but can be any system with a remote shell server.

TSO remote shell client

The TSO remote shell client enables execution of a command on a remote system with the output returned to your TSO session. The TSO remote shell can be executed from TSO and in batch. When run from batch, the output of the remote command is returned to the batch job log. The remote host need not be a z/OS system, but can be any system with a remote shell server.

9.1.3 How can remote execution be applied

Two different REXEC remote execution servers can be configured for the z/OS environment. The REXEC client command can be executed in the TSO and batch environments and in the z/OS UNIX environment. The TSO and batch environments have the same dependencies and considerations. In the REXEC TSO environment, the client can provide a user ID and password on the actual REXEC command, or a NETRC data set containing records with the appropriate user ID and password information can be used to avoid placing the security information about the command line.

Servers

You will, of course, need to implement the appropriate remote execution server or the commands that you need to process: TSO remote execution servers for TSO commands and z/OS UNIX remote execution servers for z/OS UNIX commands. If you need access to both TSO and z/OS UNIX, then both servers can be implemented at the same time in one of two ways:

- Use different ports for the z/OS UNIX servers.
- Bind the UNIX servers to a specific dynamic VIPA.

Given the current emphasis that organizations are placing on security for their environments, you might consider a client's ability to submit requests without a password a security risk. We do not recommend implementing the support in RXSERVE for an optional password; rather, we recommend requiring the user ID and the password on the command. If, however, you are required to support the optional password approach for the remote client's RSH command, then see the following documents for the detailed steps to implement that support for RXSERVE:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Security Server RACF Security Administrator's Guide*, SC28-1915
- *z/OS Security Server RACF General User's Guide*, SC28-1917

Clients

Either the TSO or the z/OS UNIX clients can be used, regardless of whether TSO, z/OS UNIX, or another platform is the server. If, however, the remote host requires user identification and access control, then you must use the TSO clients if you want to use

NETRC data sets or to submit jobs in batch. Depending on your environment, clients on your z/OS systems might need to use all three methods of remote command execution. We recommend that you become familiar with the setup and installation requirements for supporting all remote execution clients.

Roles and environments

Their roles and execution environments are positioned here:

- TSO remote execution server

  A client system might need to execute commands on another system without logging onto that target system directly. Remote execution establishes a low-level command submission and response retrieval process between two disparate systems that might otherwise not be able to communicate.

- z/OS UNIX remote execution server

  A client system might need to execute commands on another system without logging onto that target system directly. Remote execution establishes a low-level command submission and response-retrieval process between two disparate systems that, otherwise, might not be able to communicate.

  The z/OS UNIX remote execution servers can be configured to listen on any port, not just the well-known ports of 512 and 514. This provides some flexibility to your environment.

- REXEC TSO client command using user ID/password

  By specifying a user ID and password on the REXEC command itself, all of the information required by the client system and by the server system is located in one place. You avoid the clerical effort of maintaining the NETRC data set. Records in that data set must have up-to-date information related to targeted machine names, user IDs, and current passwords.

- REXEC TSO client command using the NETRC data set

  By omitting the user ID and password on the REXEC command, the plain text information is not visible on the command line being entered. You can take advantage of using a NETRC data set where you can store all the different user IDs and passwords you can have on various remote systems.

  These multiple user IDs and passwords can be maintained in one place, and the NETRC data set can be read/write protected so that only the owning user can view the contents.

- REXEC UNIX client command

  Some users of z/OS do all their work within the z/OS shell. For those users this method enables them to remain in the shell to execute the remote execution client command.

As part of any implementation effort, two appendixes in this book are beneficial in planning your work:

- Environment variables are categorized by application in Appendix A, “Environment variables” on page 395.
- Sample files for each application are listed in Appendix B, “Sample files provided with TCP/IP” on page 407.

9.2 TSO remote execution server

This section provides an overview of the TSO remote execution environment. The following topics are discussed:

- Description of TSO remote execution server
- Configuration tasks for TSO remote execution server
- Activation and verification of TSO remote execution server

9.2.1 Description of TSO remote execution server

The TSO remote execution server (RXSERVE) enables execution of a TSO command that has been received from a remote client system that is submitted by either the **rexec** or the **rsh** command.

In our environment we use the TSO REXEC client to execute a TSO command on the TSO remote execution server on another system. We also use the TSO RSH client to execute the same command on the same remote server. We show the clients executing in both batch and from TSO.

Both clients allow user ID and passwords to be specified on the command line. In addition, the REXEC client also supports the use of a NETRC data set. To illustrate the use of the NETRC data set, we use REXEC with the NETRC data set. We use RSH to illustrate use of the command line to pass the user ID and password.

TSO remote execution server dependencies

To process rexec/rsh requests from remote clients under TSO, you must execute the TSO remote execution server (RXSERVE).

A CINET environment is one in which multiple TCP/IP stacks exist on the same system image. In the context of TSO remote execution, these stacks are called *transports*. The TSO remote execution server has affinity to a specific transport in a CINET environment, so you will need to configure and execute a unique instance of the RXSERVE server for each TCP/IP stack that requires TSO remote execution services. RXSERVE determines the stack name to which it should establish affinity from the setting of the TCPIPJOBNAME variable in the TCPIP.DATA file. This file is specified in the RXSERVE JCL on the //SYSTCPD DD statement.

RXSERVE supports REXEC and RSH requests only on their well-known ports: port 512 for REXEC and port 514 for RSH. These ports cannot be configured or modified.

RXSERVE must already be actively running when a client sends a command. Otherwise, the client connection request fails.

Considerations for using TSO remote execution server

The TSO remote execution server (RXSERVE) supports both REXEC and RSH commands from a client. It does not provide flexibility to configure the two servers differently. It listens on the well-known ports only (512 and 514) and cannot be modified.

If you need to run either of the z/OS UNIX servers (orexecd or orshd) concurrently with RXSERVE, they must be configured to use a port other than their well-known ports because RXSERVE cannot be configured to listen on separate ports. However, for an alternate approach, see “Concurrent execution for TSO and z/OS UNIX remote execution servers” on page 340.

Note: When sending an RSH request to this server, the remote client can issue the RSH command without specifying a password. This server can be configured to accept or reject this type of RSH client command syntax.

If your RXSERVE server is to accept the request without a password, then you must perform a number of additional security-related tasks to ensure proper authentication and authorization for the client's request.

To support this option, the RXSERVE server must have certain surrogate job submission privileges granted by the security program on your system. See *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for the details of using the security server to define these authorizations.

9.2.2 Configuration tasks for TSO remote execution server

Perform the following tasks to implement the RXSERVE started task:

- Update the AUTOLOG and PORT statements
- Determine which remote client commands to support
- Optionally permit the RXSERVE task to be a surrogate job
- Update the RXSERVE cataloged procedure
- Create an optional user exit routine for RXSERVE
- Create one or more TSO batch procedures for RXSERVE
- Establish security access to JESSPOOL files

Update the AUTOLOG and PORT statements

If the TCP/IP stack is to start the RXSERVE task automatically when the stack initializes, include the RXSERVE name in the AUTOLOG statement in the PROFILE.TCPIP data set, as shown in Example 9-1.

Example 9-1 RXSERVE AUTOLOG statement

```
AUTOLOG
   RXSERVE ; Remote Execution Server
ENDAUTOLOG
```

Determine which remote client commands to support

Reserve port 512 for the REXEC protocol listener and port 514 for the RSH protocol listener, as shown in Example 9-2.

Example 9-2 RXSERVE port reservations

```
PORT
   512 TCP RXSERVE ; Remote Execution Server
   514 TCP RXSERVE ; Remote Execution Server
```

Optionally permit the RXSERVE task to be a surrogate job

The remote execution client can send commands to RXSERVE through port 512 by using the following methods:

- Sending the REXEC command
- Sending the RSH command with both user ID and password
- Sending the RSH command with user ID only

Our example does not support the third method. We require a password. The client commands could be as shown in Example 9-3.

Example 9-3 Client commands received by RXSERVEB

```
REXEC -l userid -p password -s 512 wtsc31.itso.ibm.com listalc
RSH -l userid/password -s 514 wtsc31.itso.ibm.com listalc
RSH -l userid -s 514 wtsc31.itso.ibm.com listalc 1
```

The RSH command in the third line **1** without a password failed on our implementation with the message in Example 9-4, because the server did not have RACF surrogate authority defined. Additionally, if your environment does not have correct surrogate job submission privileges, you might have the EZA4380E message.

Example 9-4 RSH command with missing password, error message

```
EZA4386E rshd: Permission denied.
```

Update the RXSERVE cataloged procedure

Copy the sample procedure from hlq.SEZAINST(RXPROC) to a system PROCLIB and name it RXSERVE. Modify it to meet your installation standards. Specify the parameters that RXSERVE uses to generate the batch TSO jobs that it submits.

Important: Specify two different output classes for MSGCLASS and TSCLASS. MSGCLASS must be a *held* class.

A sample RXSERVE procedure is shown in Example 9-5.

Example 9-5 Sample RXSERVEB procedure

```
BROWSE    SYS1.PROCLIB(RXSERVE) - 01.01            Line 00000000 Col 001 080
//RXSERVE PROC MODULE='RSHD',
//          EXIT=REXECEXT,
//          TSOPROC=REXECTST,
//          MSGCLASS=X,
//          TSCLASS=L,
//          MAXCONN=512,
//          PREFIX=RSHD,
//          PURGE=N,
//          IPV6=N,
//          SECLABEL=N,
//          TRACE=(LOG,NOSEND)
//*
//RXSERVE EXEC PGM=&MODULE,PARM=('EX=&EXIT,TSO=&TSOPROC',
//             'MSG=&MSGCLASS,TSC=&TSCLASS',
//             'MAX=&MAXCONN,PRE=&PREFIX,TR=&TRACE',
//             'PUR=&PURGE,IPV6=&IPV6,SL=&SECLABEL'),
//             REGION=0M,TIME=1440
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.),DISP=SHR
```

The RXSERVE parameters are explained in Example 9-6.

Example 9-6 RXSERVEB procedure JCL parameters

```
EXIT=modulename
Name of an exit routine to alter the JOB and EXEC JCL statements of the TSO batch
jobs submitted in behalf of the remote client commands.

TSOPROC=procname
Name of the TSO batch proc to be submitted in behalf of the client request.

MSGCLASS=x
Output class goes on the JOB card of the submitted job.
Must be a HELD class. Default is H.

TSCLASS=x
Output class for the //SYSTSPRT DD of the submitted job.
Must be different from MSGCLASS. Default is A.

PURGE=x (Y or N)
Should the submitted job's output be immediately purged when job completes?
Default is Y.

IPV6=x (Y or N)
Support communications with IPv6 addressing? Default is Y.

SECLABEL=x (Y or N)
Add a security label to the job card of the submitted job?
(Multilevel security related (MLS)). Default is Y.

TRACE=options
LOG | NOLOG
SEND | NOSEND
CLIENT=clientid | ALLCLIENTS
RESET (Default)

MAXCONN=nnnn
Maximum number of open sockets at any one time. Each client requires 2, Minimum =
512

PREFIX=xxxx
First four characters on the submitted JOB card, followed by a number between 1
and MAXCONN. (//XXXXn - //XXXXnnnn)
```

Create an optional user exit routine for RXSERVE

This is a optional step. The defaults that RXSERVE uses to submit the batch jobs, or even the values you specify using the JCL parameters, probably do not adhere to your installation's JCL standards. If that is the case, you must create a user exit that will adjust the parameters and values for the JOB and EXEC JCL statements. See the *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for a detailed description and sample copy of the user exit source. The source can be found in *hlq*.SEZAINST(RXUEXIT).

A sample RXUEXIT source code is shown in Example 9-7.

Example 9-7 REXECEXT routine source code

```
PARMS    DSECT
PTRINET  DC    F'0'               AF-INET or AF-INET6 socket address
PTRJOBP  DC    F'0'               Job statement parameters
PTREXECP DC    F'0'               EXEC statement parameters
PTRJES   DC    F'0'               JES control buffer
*
BUFSIZE  EQU   1024               JOB statement buffer size
*
* RXUEXIT INIT   'REXECD add class parameter to JOB statement'
* *
RXUEXIT  CSECT                    Establish the RXUEXIT csect
RXUEXIT  AMODE 31
RXUEXIT  RMODE ANY
         USING RXUEXIT,12         Establish code addressability
         STM   14,12,12(13)       Save the caller's registers
         LR    12,15              Setup the local base register
         LR    2,1                Parm pointer
         USING PARMS,2            Parameter addressability
         L     4,PTRJOBP          Job card parameters
         LR    5,4                Start of buffer
         LA    6,1                Scan 1 byte at a time
         LA    7,BUFSIZE(5)       First byte after buffer
         BCTR  7,0                Last byte to scan
SCANLOOP EQU   *
         CLI   0(5),0             Is this string termination ?
         BE    GOTEND             Yes
         BXLE  5,6,SCANLOOP       Continue scan for term
* ---------------------------------------------------------------
* If string is not null terminated, return without altering
* ---------------------------------------------------------------
         B     RETURN             Should not happen.
GOTEND   EQU   *
         LR    6,5                address of null byte
         SR    5,4                L'job parameter statements
         LA    5,L'CLASS(5)       New parameter length
         CH    5,=AL2(BUFSIZE)    Do we exceed buffer size?
         BNH   LENOK              No, there is room enough
* ---------------------------------------------------------------
* String length would exceed buf size so return without altering
* ---------------------------------------------------------------
         B     RETURN             Return without modification
*
LENOK    EQU   *
         MVC   0(L_CLASS,6),CLASS  Move class statement to buff
         L     6,PTRJES           Get address of JES buffer
         MVC   0(L_JES2,6),JES2CNTL Move JES2 control to buffer
RETURN   EQU   *
         LM    14,12,12(13)       Restore the registers
         LA    15,0(0,0)          Load the return code
         BR    14                 Return
*
         LTORG
```

```
*
CLASS    DC    C',CLASS=L'         Class statement           1
         DC    X'00'               null termination byte
L_CLASS  EQU   *-CLASS
*
JES2CNTL DC    C'/*JOBPARM SYSAFF=SC31' JES2 system affinity  2
         DC    X'00'                    null termination byte
L_JES2   EQU   *-JES2CNTL
*
JES3CNTL DC    C'//*MAIN SYSTEM=(MAIN1)' JES3 main assignment
         DC    X'00'                      null termination byte
L_JES3   EQU   *-JES3CNTL
*
         END
```

In this example, the numbers correspond to the following information:

1. The CLASS statement in user routine should be assigned to the same value as the TSCLASS parameter specified in RXSERVE start procedure.
2. Specify the system affinities if your environment require it. If you remote access a z/OS image in a sysplex and there are different multiple naming convention TCP/IP stacks, this parameter can let your remote access commands go to the right system, otherwise the connection to TCP/IP stack could fail.

Important: Before assigning a HOLD class for the MSGCLASS in RXSERVE start procedure, use the JES2 command $D OUTCLASS to ensure that the OUTDISP of this outclass is OUTDISP=(HOLD,HOLD).

Otherwise, the job output will not be returned to the remote user, and the user will receive a `Jobname not found` message when the job times out. In addition, the MSGCLASS cannot be modified by the MODIFY command.

The user exit should have the AMODE(31) and RMODE(24) attributes to provide addressability to the input parameters. Example 9-8 shows the JCL to assemble and link-edit the RXUEXIT source code.

Example 9-8 Assembling and linking RXUEXIT

```
//ASMLINK  JOB ,,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,
//             REGION=0M,MSGLEVEL=(1,1)
//ASM     EXEC PGM=ASMA90,PARM='DECK,NOOBJECT,LIST'
//SYSIN    DD  DISP=SHR,DSN=TCPIPB.TCPPARMS(RXUEXIT)
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//         DD  DISP=SHR,DSN=SYS1.AMODGEN
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPUNCH DD  DSN=&&LOADSET,UNIT=SYSDA,DISP=(,PASS),
//             SPACE=(400,(100,100,1)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//SYSPRINT DD  SYSOUT=*
//LKED    EXEC PGM=HEWL,COND=(4,LT,ASM),
//             PARM='XREF,AMODE=31,RMODE=24'
//SYSLMOD  DD  DSN=SYS1.COMMPLEX.LINKLIB(REXECEXT),DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
```

The SYSLMOD data set should have APF authorization and in LINKLST. After get return code ‘0’ from this job, refresh LLA before using this user exit routine.

Create one or more TSO batch procedures for RXSERVE

When using the RXSERVE server, the procedure specified in the TSOPROC parameter of the startup procedure for RXSERVE must have the //SYSTSPRT DD statement appearing before any other output DD specifications in the procedure. RXSERVE uses the Job Entry Subsystem (JES) interface to select the first file of the first output group from the job.

Example 9-9 shows the correct placement of the //SYSTSPRT DD statement within the JCL, if the batch procedure specified is TSOPROC=REXECTST.

Example 9-9 Sample TSOPROC procedure

```
BROWSE    SYS1.PROCLIB(REXECTST) - 01.03            Line 00000000 Col 001 080
//REXECTST PROC
//GENERAL  EXEC PGM=IKJEFT01,DYNAMNBR=90,TIME=1440
//STEPLIB  DD DSN=ISP.SISPLOAD,DISP=SHR
//SYSPROC  DD DSN=ESA.SYS1.CLIST,DISP=SHR
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB31),DISP=SHR
//SYSTSPRT DD TERM=TS,SYSOUT=*
//SYSPRINT DD TERM=TS,SYSOUT=*
//SYSTERM  DD TERM=TS,SYSOUT=*
//*
```

RXSERVE overrides the class specification for the //SYSTSPRT DD statement with the value specified on the TSCLASS parameter before it submits the job. RXSERVE also adds a //SYSTSIN DD statement to the TSOPROC procedure, where it includes the TSO command requested by the client on the original REXEC or RSH command. Example 9-3 on page 333 shows what a client request looks like.

Establish security access to JESSPOOL files

The submitted TSO commands normally generate output. This output is placed onto the JES spool through the //SYSTSPRT DD JCL statement. RXSERVE pulls this output from the JES spool and returns it to the requesting client host. If the user IDs associated with the batch jobs submitted by RXSERVE require JESSPOOL access authority, use the IBM RACF security server or equivalent System Authorization Facility (SAF) interface to set the required level of authority. Security details can be found in:

- *z/OS Security Server RACF Security Administrator’s Guide*, SC28-1915
- *z/OS Security Server RACF General User’s Guide*, SC28-1917

9.2.3 Activation and verification of TSO remote execution server

RXSERVE can be started by an automations package, by a manual command (S RXSERVE), or by TCP/IP at initialization of the TCP/IP address space by way of the AUTOLOG statement within the PROFILE data set. When RXSERVE is started, the system issues the messages shown in Example 9-10.

Example 9-10 RXSERVE startup messages

```
$$HASP100 RXSERVE ON STCINRDR
IEF695I START RXSERVE WITH JOBNAME RXSERVE IS ASSIGNED TO USER TCPIP GROUP TCPGRP
$HASP373 RXSERVE  STARTED
EZA4400I Trace options: LOG,SEND,ALLCLIENTS
```

```
EZA4404I Parameters:
MSGCLASS=X,TSCLASS=L,TSOPROC=REXETEST,MAXCONN=512,EXIT=REXECEXT
EZA4423I Parameters: PREFIX=RSHD,PURGE=N,IPV6=N,SECLABEL=N
Established affinity with TCPIPB 1
EZA4414I rexecd: Initialization using JES=2 selected by Automatic Selection
descarray is at 13550160, size is 8 bytes
descarray has 512 entries, entry size is 788
```

In this example, the TCP/IP stack affinity is specified in TCPDATA on SYSTCPD DD statement **1**.

9.3 z/OS UNIX remote execution server

This section provides an overview of the UNIX remote execution server (orexecd/rexecd) environment. The following topics are discussed:

- Description of z/OS UNIX remote execution server
- Configuration tasks for z/OS UNIX remote execution server
- Activation and verification of z/OS UNIX remote execution server

9.3.1 Description of z/OS UNIX remote execution server

The UNIX remote execution server (orexecd/rexecd) and remote shell server (orsh/rsh) enables execution of a z/OS UNIX command that has been received from a remote client host using the **rexec** or **rsh** command.

In our environment we use the z/OS orexec client to execute a z/OS UNIX command on the z/OS orexecd server on another system. We also use the z/OS UNIX orsh client to execute the same command on the same remote server.

z/OS UNIX remote execution dependencies

To process **orexec/rexec** requests from remote clients under z/OS UNIX, you must execute the z/OS UNIX remote execution server (orexecd/rexecd). Likewise, to process orsh/rsh requests from remote clients under z/OS UNIX, you must execute the z/OS UNIX remote shell server (orshd/rshd).

The z/OS UNIX servers are initiated through the INETD server and can be configured to use ports other than the well-known ports, 512 and 514. If INETD and the servers have been configured to serve different ports, the client must be aware of this and specify that specific port on the request.

INETD listens on the configured port on behalf of the z/OS UNIX remote execution servers. INETD initiates an instance of the server at the time a client request is received. The server is not activated ahead of time. INETD must be listening on the appropriate port, and the client must specify the same port—otherwise the client connection request fails.

For rexec requests, INETD listens on the port number that is specified for *exec* in the `/etc/services` file. For rsh requests, INETD listens on the port number that is specified for “shell”. Make sure the intended port numbers are specified.

Considerations for using z/OS UNIX remote execution

The z/OS UNIX remote execution server (orexecd/rexecd) services only the REXEC, **orexec**, and **rexec** commands from a client. To service RSH, **orsh**, and **rsh** clients, the RSH server (orshd/rshd) must also be executed.

If you need to run the z/OS UNIX server (orexecd/rexecd) concurrently with RXSERVE, the z/OS UNIX server must be configured to support a port other than its well-known port of 512 because RXSERVE cannot be modified. For details about running both servers concurrently, see “Concurrent execution for TSO and z/OS UNIX remote execution servers” on page 340.

The z/OS UNIX remote execution clients cannot use a NETRC data set, and jobs cannot be submitted in batch.

9.3.2 Configuration tasks for z/OS UNIX remote execution server

The z/OS UNIX System Services remote execution server, orexecd, enables UNIX commands to be submitted from a remote host and executed on the local z/OS system. The INETD server listens on the designated port on behalf of the orexecd server. When a remote request comes in on that port from a remote client host, INETD initiates an instance of the orexecd server to process the inbound request. The orexecd server is identified by the service name of exec in `/etc/services`. Place the **exec** command, along with appropriate parameters, into the `/etc/inetd.conf` file.

The orexecd record in the INETD `/etc/inetd.conf` file could look like one of the entries in Example 9-11, depending on the options that might be desired. See the *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for a complete description of all the parameters available to rexecd.

Example 9-11 Sample /etc/inetd.conf file records for rexecd

```
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd -LV
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd -s
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd -d -l -v -c -s
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd
```

In our scenario, the actual record for **rexecd** 1 in an `/etc/inetd.conf` file is shown in Example 9-12.

Example 9-12 Sample /etc/inetd.conf file with orexecd/rexecd record

```
#=====================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV   1
echo     stream tcp nowait OMVSKERN internal
discard  stream tcp nowait OMVSKERN internal
chargen  stream tcp nowait OMVSKERN internal
daytime  stream tcp nowait OMVSKERN internal
time     stream tcp nowait OMVSKERN internal
echo     dgram  udp wait   OMVSKERN internal
discard  dgram  udp wait   OMVSKERN internal
```

```
chargen  dgram  udp wait   OMVSKERN internal
daytime  dgram  udp wait   OMVSKERN internal
time     dgram  udp wait   OMVSKERN internal
pop3     stream tcp nowait bpxroot  /usr/sbin/popper popper
```

Make certain that the *exec* entry in `/etc/services` indicates the intended port number over which you want rexecd to process requests. INETD will listen on behalf of orexecd/rexecd on the port specified in `/etc/services`. The default well-known port for exec/rexecd/orexecd is 512 **1**. Notice the default port for shell/rsh/orsh is 514 **2**, as shown in Example 9-13.

Example 9-13 /etc/services entries for exec and shell

```
#
# UNIX specific services
#
exec            512/tcp 1
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp 2       cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
```

9.3.3 Activation and verification of z/OS UNIX remote execution server

When INETD is up and active, use a NETSTAT/**onetstat** command to verify that it is listening on the port specified for exec/rexecd/orexecd **1**, as shown in Example 9-14.

Example 9-14 NETSTAT display shows INETD listening on port 512

```
User Id  Conn     State
-------  ----     -----
INETD1   00000040 LISTEN
  LOCAL SOCKET:   0.0.0.0..512   1
  FOREIGN SOCKET: 0.0.0.0..0
INETD1   00000040 LISTEN
  LOCAL SOCKET:   0.0.0.0..514
  FOREIGN SOCKET: 0.0.0.0..0
```

Concurrent execution for TSO and z/OS UNIX remote execution servers

We previously mentioned that one way to concurrently execute both TSO and z/OS UNIX servers is to change the port numbers for the z/OS UNIX server in `/etc/services`. It could listen on a different port and not conflict with the TSO server. However, that would be confusing to remote clients that might require using the well-known port for submitting requests.

Another method can be used to achieve concurrent execution for the two servers. Both servers are generic listeners, or generic servers. We can make one of the servers appear to be a BIND-specific server and avoid the port conflict.

Because the remote execution servers are generic servers, they attempt to bind to INADDR_ANY when they are started, which allows them to listen on all defined IP addresses. However, this bind prevents both the TSO and z/OS UNIX remote execution servers from

listening on the same port, and one of the servers has to use a nonstandard port. Using the BIND parameter on the PORT reservation statement in the TCPIP profile data set allows both the TSO and z/OS UNIX remote execution servers to bind to the same ports using different IP addresses. The following steps illustrate how this can be done.

1. Define a VIPA address (either static or dynamic, but preferably dynamic) to the PROFILE.TCPIP data set with a pair of DEVICE and LINK statements (or VIPADYNAMIC statements) and add the LINK to the HOME statement for static VIPA. We used Dynamic VIPA and VIPARANGE statement in our setup. This is shown in Example 9-15.

Example 9-15 VIPA for BIND of `orexecd` and orsh

```
DEVICE VIPA1    VIRTUAL 0
 LINK  VIPA1L   VIRTUAL 0 VIPA1
;
   HOME
      10.1.4.21 IUTIQDF4L
      10.1.5.21 IUTIQDF5L
      10.1.6.21 IUTIQDF6L
      10.1.1.20 VIPA1L
      10.1.2.21 OSA2080L
      10.1.2.22 OSA20A0L
      10.1.3.21 OSA20C0L
      10.1.3.22 OSA20E0L
   ;
OR
   VIPADYNAMIC
    ;-------------------------------------------------------------------
    ;  Set aside some addresses for use with BIND and SIOCSVIPA IOCTL   -
    ;              (10.1.9.21 thru 10.1.9.24)                          -
    ;-------------------------------------------------------------------
      VIPARANGE  DEFINE  255.255.255.255 10.1.9.21
      VIPARANGE  DEFINE  255.255.255.255 10.1.9.22
      VIPARANGE  DEFINE  255.255.255.255 10.1.9.23
      VIPARANGE  DEFINE  255.255.255.255 10.1.9.24
   ENDVIPADYNAMIC
```

2. Add PORT statements to the TCPIP profile for both the TSO and z/OS UNIX remote execution servers. One of the servers will bind to the VIPA address. The other can bind to INADDR_ANY by not specifying the BIND parameter. In this case, the z/OS UNIX remote execution servers bind to the VIPA address, as shown in Example 9-16 on page 342.

Important: The server *with* the BIND parameter must be listed *before* the one without the BIND parameter. This setup directs all requests to ports 512 or 514 with a destination IP address of 10.1.9.22 to the z/OS UNIX remote execution servers.

Requests to ports 512 or 514 with a destination IP address that is not 10.1.9.22 are directed to the TSO remote execution server.

Example 9-16 BIND-specific assignments for orexecd and orshd

```
PORT
...
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.1.9.21 ; OE Telnet Server D-VIPA
...
;  443 UDP HTTPS               ; http protocol over TLS/SSL
   512 TCP OMVS BIND 10.1.9.22           ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.1.9.22           ; UNIX RSHD   D-VIPA
   512 TCP RXSERVE                       ; TSO  REXECD
   514 TCP RXSERVE                       ; TSO  RSHD
   515 TCP LPSERVEB
...
```

3. Verify in the `/etc/services` file that exec uses port 512 and shell uses 514.
4. Verify the setup by starting the stack with the new changes and starting RXSERVE and INETD, the two listeners involved.
5. Issue the NETSTAT command, and it should now show that both REXECD servers are listening on port 512 **1** and both RSH servers are listening on port 514 **2**. INETD is now listening for the z/OS UNIX remote execution servers, in this case, on the VIPA address only **3** as shown in Example 9-17.

Example 9-17 NETSTAT verifies BIND-specific listeners

```
User Id  Conn     State
-------  ----     -----
INETD1   00000040 LISTEN
  Local Socket:   10.1.9.22..512 1 3
  Foreign Socket: 0.0.0.0..0
INETD1   0000009D LISTEN
  Local Socket:   10.1.9.22..514 2 3
  Foreign Socket: 0.0.0.0..0
RXSERVE  000000D4 LISTEN
  LOCAL SOCKET:   0.0.0.0..512   1
  FOREIGN SOCKET: 0.0.0.0..0
RXSERVE  000000D5 LISTEN
  LOCAL SOCKET:   0.0.0.0..514   2
  FOREIGN SOCKET: 0.0.0.0..0
```

For more information about the PORT reservation statement, the DEVICE and LINK statements, and the HOME statement, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

9.4 REXEC TSO client command using user ID/password

This section provides an overview of the REXEC TSO client command with a user ID and password specified. The following topics are discussed:

- Description of REXEC TSO with user ID and password
- Configuration of REXEC TSO with user ID and password
- Verification of REXEC TSO with user ID and password

9.4.1 Description of REXEC TSO with user ID and password

The client can specify a user ID and password on the REXEC command line. This user ID and password pair is the user ID that the remote system is to use for authorizing execution of the requested command.

Dependencies of REXEC TSO with user ID and password

To use REXEC, a REXEC daemon must be running on the remote host. The REXEC client passes the user name, password, and command to the remote REXEC daemon. The daemon provides automatic logon and user authentication, depending on the parameters that you set.

The user ID specified on the REXEC command must be a valid user ID on the remote targeted system.

Considerations for using REXEC TSO with user ID and password

By specifying a user ID and password on the REXEC command itself, the information is transmitted along with the command to the remote system in plain text.

9.4.2 Configuration of REXEC TSO with user ID and password

This section shows how to use the REXEC TSO client command with a user ID and password specified. Use the REXEC command to execute a command on a remote host and receive the results on the local system, as shown in Example 9-18.

Example 9-18 REXEC TSO client command format

```
REXEC ? -b tab -d -m -n -l userid -p password -s portnum -t translate_file
foreign_host command
```

The *foreign_host* and the *command* are both required. All other parameters are optional. However, because this example is discussing the use of a user ID and password on the REXEC command, we assume that those two parameters are included on the command for this example. The remote-host can be a DNS name to be resolved by DNS services, or it can be the IP address of the remote host.

The parameters **-b**, **-d**, **-l**, **-m**, **-n**, **-p**, **-s**, and **-t** are case sensitive and must be entered in lowercase letters. The user ID and password values might be case sensitive, depending on the remote operating system requirements.

Use a question mark (?) to obtain command help, as shown in Example 9-19.

Example 9-19 REXEC TSO client command help

```
===> REXEC ?
 rexec: no command given.
 Usage: rexec  -? -d -b <tab> -l <usr> -p <pwd> -n -m -s <port> -t <fn> Fhost c
 md
  options: -
           -?          display this message.
           -d          turn on debug tracing.
           -m          prefix '09'x to machine control output.
           -b <tab>    specifies tab value,
                       valid range 1 - 12 (default 1).
           -l <usr>    specifies remote userid.
```

```
          -p <pwd>    specifies remote password.
          -n          prevent automatic login.
          -s <port>   specifies server port (default 512).
          -t <fn>     specifies translation table name.

  Example: rexec  -d -l guest -p guest hostname ls
```

Available parameters include:

- Use **-b tab** to specify the Tab setting. Valid values are in the range 1 - 12 and the default value is 1.
- Use **-d** to activate debug tracing.
- Use **-m** to specify that the machine control character (X'09') is added to the beginning of the output lines for the data sets that are associated with the SYSPRINT or OUTPUT DD cards and have the machine control attribute. When you use this parameter, it should be the first parameter that is passed to REXEC so that all output lines are changed.
- Use **-n** to prevent an automatic logon, and to force a password prompt to the client. This option is applicable only when the NETRC data set has been used to obtain the user ID and password. It prevents the client support function from automatically using the obtained password, and forces a password prompt to the client. This example does not consider the **-n** parameter because we are assuming the use of a user ID and password on the REXEC command.
- Use **-l** to specify the user ID to be used by the remote host.
- Use **-p** to specify the password for the user ID on the remote host.
- Use **-s** to specify the TCP port number of the REXEC server on the remote host. The port number specified here must match the port number on which the remote REXEC server is listening. The default well-known port is 512.
- Use **-t** to override the default translation table name. A search order process is used by the system to determine which translation table to use, as shown in Example 9-20.

Example 9-20 REXEC TSO client command translation table search order

```
If no translation table name is specified by using the -t parameter, search for:
   userid.STANDARD.TCPXLBIN
   hlq.STANDARD.TCPXLBIN where hlq is specified by DATASETPREFIX in the TCPDATA
   file

If the standard table is not present, then a hardcoded default table identical to
hlq.SEZATCPX(STANDARD) is used.

If -t specifies a translation table of -t translate_file, search for:
   userid.translate_file.TCPXLBIN
   hlq.translate_file.TCPXLBIN

If the specified file is not found, REXEC terminates with message EZA4805I
```

REXEC TSO requests can be submitted from the TSO command line or from a batch job. These methods include:

- Submitting REXEC TSO requests including a user ID and password
- Submitting REXEC TSO requests in batch with a user ID and password

Submitting REXEC TSO requests including a user ID and password

Use the REXEC command with a user ID and password specified, requiring no NETRC data set, as shown in Example 9-21.

Example 9-21 REXEC TSO client command with a user ID and password

```
READY
rexec -l CS07 -p PASSWORD 10.1.1.20 netstat home
```

Submitting REXEC TSO requests in batch with a user ID and password

You usually run REXEC interactively by entering the command and then receiving the results at your terminal. However, you can also run REXEC as a batch job. To accomplish this, you must supply the necessary job control language (JCL) and submit it to the Job Entry Subsystem (JES) using the TSO SUBMIT command. The command format when submitted as a batch job is the same as the command format for TSO, described in Example 9-18 on page 343.

The command can be entered as a parameter on the EXEC JCL statement, as shown in Example 9-22.

Example 9-22 Batch REXEC TSO client command using EXEC PARM=

```
BROWSE    CS07.TCPPARMS(JOBRX1) - 01.04             Line 00000000 Col 001 080
//CS07Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS07
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='-l CS07 -p PASSWORD 10.1.1.20 netstat home'
//SYSPRINT DD DSN=CS07.NETSTAT.HOME,DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB31)
```

9.4.3 Verification of REXEC TSO with user ID and password

If you omit the user ID, the password, or both when entering the REXEC command, *and* if the remote host is not specified with a user ID or password in the NETRC data set, then the local system prompts the client for the missing parameters.

Note: The data set containing the JCL cannot have sequence numbers.

You can verify the results of the REXEC TSO client command by reviewing the response that you get on your TSO session, as shown in Example 9-23.

Example 9-23 Results of REXEC TSO client command from a TSO session

```
Home address list:
LinkName:   IUTIQDF4L
  Address:  10.1.4.21
    Flags:
LinkName:   IUTIQDF5L
  Address:  10.1.5.21
    Flags:
LinkName:   IUTIQDF6L
  Address:  10.1.6.21
    Flags:
LinkName:   VIPA1L
  Address:  10.1.1.20
    Flags:  Primary
```

```
LinkName:   OSA2080I
  Address:  10.1.2.21
    Flags:
IntfName:   OSA20A0I
  Address:  10.1.2.22
    Flags:
IntfName:   OSA20C0I
  Address:  10.1.3.21
.......................
***
```

You can verify the results of the batch job REXEC TSO client command by reviewing the response it places in the output data set specified in the batch JCL, as shown in Example 9-24.

Example 9-24 Results of REXEC TSO client command from a batch job

```
BROWSE    CS07.NETSTAT.HOME                          Line 00000000 Col 001 080
READY
netstat home
Home address list:
LinkName:   IUTIQDF4L
  Address:  10.1.4.21
    Flags:
LinkName:   IUTIQDF5L
  Address:  10.1.5.21
    Flags:
LinkName:   IUTIQDF6L
  Address:  10.1.6.21
    Flags:
LinkName:   VIPA1L
  Address:  10.1.1.20
    Flags:  Primary
LinkName:   OSA2080I
  Address:  10.1.2.21
    Flags:
LinkName:   OSA20A0I
  Address:  10.1.2.22
    Flags:
LinkName:   OSA20C0I
  Address:  10.1.3.21
    Flags:
LinkName:   OSA20E0I
  Address:  10.1.3.22
    Flags:
.......................
READY
END
```

9.5 REXEC TSO client command using the NETRC data set

This section provides an overview of the REXEC TSO client command with a NETRC data set used to obtain user ID and password information. The following topics are discussed:

- Description of REXEC TSO client using NETRC
- Configuration of REXEC TSO client using NETRC
- Verification of REXEC TSO client using NETRC

9.5.1 Description of REXEC TSO client using NETRC

The client can omit the user ID and password information from the REXEC command line. The client function of TSO locates a NETRC data set, and retrieves the user ID and password information from an appropriate record within that data set. The user ID that is located is passed on the remote system for authorizing execution of the requested command.

Dependencies of REXEC TSO client using NETRC

To use REXEC, a REXEC daemon must be running on the remote host. The REXEC client passes the user name, password, and command to the remote REXEC daemon. The daemon provides automatic logon and user authentication, depending on the parameters that you set. The user ID specified on the REXEC command must be a valid user ID on the remote targeted system.

A NETRC data set must exist and be configured with appropriate machine, user ID, and password information.

Considerations for REXEC TSO client using NETRC

The NETRC data set must be maintained and kept up to date with current passwords for the user on each targeted machine involved.

If the NETRC data set is needed because the user ID and password have been omitted from the client command, and it is not found by the client support function, the client request fails.

9.5.2 Configuration of REXEC TSO client using NETRC

This section shows how to use the REXEC TSO client command, omitting the user ID and password from the command and acquiring them from the NETRC data set. Use the REXEC (TSO client) command to execute a command on a remote host and receive the results on the local system, as shown in Example 9-25.

Example 9-25 REXEC command format with no user ID or password: for use with NETRC data set

```
REXEC ? -b tab -d -m -n -s portnum -t translate_file foreign_host command
```

REXEC TSO requests can be submitted from the TSO command line or from a batch job. These methods include:

- Submitting REXEC TSO client requests without a user ID and password
- Submitting REXEC TSO client requests in batch without user ID and password
- Preparing and using the NETRC data set

Submitting REXEC TSO client requests without a user ID and password

Use the REXEC command without a user ID and password specified, requiring the NETRC data set, as shown in Example 9-26.

Example 9-26 REXEC TSO client command without a user ID and password

```
READY
rexec 10.1.1.20 netstat conn
```

Submitting REXEC TSO client requests in batch without user ID and password

You usually run REXEC interactively by entering the command and then receiving the results at your terminal. However, you can also run REXEC as a batch job. To accomplish this, you must supply the necessary job control language (JCL) and submit it to the Job Entry Subsystem (JES) using the TSO SUBMIT command. The command format when submitted as a batch job is the same as the command format described in “Submitting REXEC TSO requests including a user ID and password” on page 345.

The command is entered as a parameter on the EXEC JCL statement. The results of the command executed on the remote host are stored on the local host and by default written to the //SYSPRINT DD allocation. The data set characteristics should be consistent with the output from the command you are executing at the remote host. This is shown in Example 9-27.

Example 9-27 Batch REXEC TSO client command without a user ID and password

```
BROWSE    CS07.TCPPARMS(JOBRX3) - 01.03             Line 00000000 Col 001 080
//CS07Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS07
//STEP1    EXEC PGM=REXEC,REGION=0M,
//     PARM='10.1.1.20 netstat home'
//SYSPRINT DD SYSOUT=*
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
```

Preparing and using the NETRC data set

The NETRC data set provides an alternative to specifying the user ID and password as REXEC parameters. The following NETRC discussion applies to the interactive TSO environment and to the batch job environment. It is clearer to use the batch job environment to illustrate the use of the NETRC data set.

REXEC uses the following search order to find the NETRC data set:

1. //NETRC DD statement
2. *userid*.NETRC.DATA
3. *tso_prefix*.NETRC
4. *userid*.NETRC

Note: NETRC characteristics to be aware of are:

- If the password is specified using the -p parameter on the REXEC client command, no NETRC data set is used. Otherwise, the NETRC data set is used to acquire the password.
- If the password is omitted from both the REXEC client command and the machine record within the NETRC data set, the REXEC program prompts you for your current password. You must be running in an interactive TSO session.
- If you have submitted a batch job to execute the REXEC client command, and you omit the password from both the REXEC client command and from the machine record within NETRC, then the REXEC client command fails because it cannot prompt for a password in batch.

The format of the records within the NETRC data set is shown in Example 9-28.

Example 9-28 NETRC record syntax

```
machine remotehost login userid password userpswd
```

The keywords *machine*, *login*, and *password* must be specified in lowercase, exactly as they are spelled in the example. The remotehost specification can be its DNS name or its IP address. The user ID and password might be case sensitive at the remote host, and if supplied in the incorrect case, failure might occur when connecting to a REXEC server.

Example 9-29 shows the NETRC data set that we used in our scenarios. It is a RECFM=FB, LRECL=80 sequential data set.

Example 9-29 Sample NETRC data set: cs07.NETRC

```
MACHINE 127.0.0.1                       LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC30.ITSO.IBM.COM             LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC31.ITSO.IBM.COM             LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC32.ITSO.IBM.COM             LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC33.ITSO.IBM.COM             LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.10                       LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.20                       LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.30                       LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.40                       LOGIN CS07     PASSWORD CS07PSWD
```

When running REXEC in batch, the user ID assigned to the batch job is used as the high-level qualifier in locating the default *userid*.NETRC.DATA or the *userid*.NETRC data set. Example 9-30 shows the use of the *userid*.NETRC.DATA data set containing the user ID and password for the user ID assigned to the batch job. The output is sent to the data set indicated on the //SYSPRINT DD statement.

Example 9-30 Batch REXEC TSO client command JCL using default NETRC data set

```
BROWSE    CS07.TCPPARMS(JOBRX3) - 01.03              Line 00000000 Col 001 080
//JOBRX3 JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=&SYSUID
//STEP1    EXEC PGM=REXEC,REGION=0M,
//     PARM='10.1.1.20 netstat conn'
//SYSPRINT DD SYSOUT=*
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
```

The //NETRC DD statement can be used in the batch job to override the default search order. Example 9-31 shows the use of the //NETRC DD statement in batch.

Example 9-31 Batch REXEC TSO client command JCL overriding default NETRC data set

```
//RXTEST JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=&SYSUID
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='10.1.1.20 netstat conn'
//SYSPRINT DD SYSOUT=*
//NETRC    DD   DSN=CS07.OVERRIDE.NETRC,DISP=SHR
```

9.5.3 Verification of REXEC TSO client using NETRC

If you omit the user ID, the password, or both when entering the REXEC command, *and* if the remote host is not specified with a user ID or password in the NETRC data set, then the local system prompts the client for the missing parameters.

Note: The data set containing the JCL cannot have sequence numbers.

You can verify the results of the REXEC TSO client command by reviewing the response you get back on your TSO session, as shown in Example 9-32.

Example 9-32 Results of REXEC TSO client command without user ID and password in TSO

```
User Id  Conn     State
-------  ----     -----
DCD1DIST 0000225A Listen
  Local Socket:   0.0.0.0..38241
  Foreign Socket: 0.0.0.0..0
DCD1DIST 00002257 Listen
  Local Socket:   0.0.0.0..38240
  Foreign Socket: 0.0.0.0..0
OMPB     00000026 Establsh
  Local Socket:   127.0.0.1..1026
  Foreign Socket: 127.0.0.1..1027
TCPIPB   00000017 Listen
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 0.0.0.0..0
TCPIPB   0000001D Establsh
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 127.0.0.1..1025
TCPIPB   00000025 Establsh
  Local Socket:   127.0.0.1..1027
  Foreign Socket: 127.0.0.1..1026
TCPIPB   0000001C Establsh
  Local Socket:   127.0.0.1..1025
  Foreign Socket: 127.0.0.1..1024
TN3270B  00000034 Listen
  Local Socket:   ::..23
  Foreign Socket: ::..0
TN3270B  00000033 Listen
  Local Socket:   ::..992
  Foreign Socket: ::..0
TCPIPB   000023A9 UDP
  Local Socket:   ::..3512
  Foreign Socket: *..*
***
```

An example of the batch job that RXSERVE submits (using the TSOPROC specified in the RXSERVE parms) is shown in Example 9-33. The originating user job did not specify a user ID or password on the REXEC command, so the REXEC program uses the default NETRC data set because the //NETRC DD statement was not supplied in the originating JCL.

Example 9-33 REXEC using default NETRC when no user ID or password specified on command

```
SDSF OUTPUT DISPLAY CS07Z    JOB07179  DSID     2 LINE 0       COLUMNS 02- 81
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 0  --  N O D E

23.22.51 JOB07179  IRR010I  USERID CS07     IS ASSIGNED TO THIS JOB.
23.23.25 JOB07179  ICH70001I CS07     LAST ACCESS AT 23:19:07 ON THURSDAY,
September
23.23.25 JOB07179  $HASP373 CS07Z    STARTED - INIT 3    - CLASS C - SYS SC30
        1 //CS07Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS0
        2 //STEP1    EXEC PGM=REXEC,REGION=0M,
          //     PARM='-d 10.1.1.20 netstat home'
        3 //SYSPRINT DD SYSOUT=*
        4 //SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
ICH70001I CS07     LAST ACCESS AT 23:19:07
IEF236I ALLOC. FOR CS07Z STEP1
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I 803B ALLOCATED TO SYSTCPD
IEF237I 803B ALLOCATED TO SYS00004
Established affinity with TCPIPB
EZA4809I  Using NETRC file //'CS07.NETRC'.
EZA4762I  MACHINE : 10.1.1.20
EZA4763I  LOGIN   : CS07
EZA4764I  PASSWORD: ******
EZA4801I  MVS TCP/IP REXEC CS
EZA4775I  Calling function rexec_af with the following:
Host: 10.1.1.20   user: CS07   cmd: netstat home   port: 512
Getaddrinfo successful
EZA4774I  rexec invoked;
Data socket = 1  Control socket = 3
```

The job that was submitted by RXSERVE to execute the requested command under TSO batch is shown in Example 9-34.

Example 9-34 TSOPROC job submitted by RXSERVE (REXECTST)

```
SDSF OUTPUT DISPLAY CS074    JOB07180  DSID     2 LINE 0       COLUMNS 02- 81
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 0  --  N O D E

23.22.57 JOB07180  ICH70001I CS07     LAST ACCESS AT 23:22:57 ON THURSDAY,
September
23.22.57 JOB07180  $HASP373 CS074    STARTED - INIT 2    - CLASS A - SYS SC30
23.22.57 JOB07180  -                                        ---------TIMINGS (M
23.22.57 JOB07180  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB   VECT
23.22.57 JOB07180  -CS074    TSO      GENERAL     00    466    .00    .00    .00
23.22.57 JOB07180  -CS074    ENDED.  NAME-                     TOTAL CPU TIME=
23.22.57 JOB07180  $HASP395 CS074    ENDED
------ JES2 JOB STATISTICS ------
  21 Sep 2007 JOB EXECUTION DATE
           11 CARDS READ
          108 SYSOUT PRINT RECORDS
```

```
            0 SYSOUT PUNCH RECORDS
            4 SYSOUT SPOOL KBYTES
         0.00 MINUTES EXECUTION TIME
        1 //CSO74    JOB CSO7,
          // USER=CSO7,
          // PASSWORD=,
          // MSGCLASS=A
        2 //TSO EXEC REXECTST
        3 XXREXECTST PROC
        4 XXGENERAL  EXEC PGM=IKJEFT01,DYNAMNBR=90,TIME=1440
        5 XXSTEPLIB  DD DSN=ISP.SISPLOAD,DISP=SHR
        6 XXSYSPROC  DD DSN=ESA.SYS1.CLIST,DISP=SHR
        7 XXSYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB30),DISP=SHR
        8 //SYSTSPRT DD SYSOUT=(H,RSHD),HOLD=YES
          X/SYSTSPRT DD TERM=TS,SYSOUT=*
        9 //SYSPRINT DD RECFM=VBA,SYSOUT=(*,RSHD),HOLD=YES
          X/SYSPRINT DD TERM=TS,SYSOUT=*
       10 //SYSTERM  DD RECFM=VBA,SYSOUT=(*,RSHD),HOLD=YES
          X/SYSTERM  DD TERM=TS,SYSOUT=*
          XX*
       11 //DEFAULT  OUTPUT DEFAULT=YES,CONTROL=PROGRAM,WRITER=RSHD
       12 //SYSTSIN  DD *
 STMT NO. MESSAGE
```

You can verify the results of the batch job REXEC TSO client command by reviewing the response it places into the output data, as shown in Example 9-35.

Example 9-35 Results of REXEC TSO client command without user ID and password in batch

```
BROWSE    CSO7.NETSTAT.CONN                          Line 00000000 Col 001 080
User Id  Conn     State
-------  ----     -----
DCD1DIST 0000225A Listen
  Local Socket:   0.0.0.0..38241
  Foreign Socket: 0.0.0.0..0
DCD1DIST 00002257 Listen
  Local Socket:   0.0.0.0..38240
  Foreign Socket: 0.0.0.0..0
OMPB     00000026 Establsh
  Local Socket:   127.0.0.1..1026
  Foreign Socket: 127.0.0.1..1027
TCPIPB   00000017 Listen
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 0.0.0.0..0
TCPIPB   0000001D Establsh
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 127.0.0.1..1025
TCPIPB   00000025 Establsh
  Local Socket:   127.0.0.1..1027
  Foreign Socket: 127.0.0.1..1026
TCPIPB   0000001C Establsh
  Local Socket:   127.0.0.1..1025
  Foreign Socket: 127.0.0.1..1024
TN3270B  00000034 Listen
  Local Socket:   ::..23
  Foreign Socket: ::..0
```

```
TN3270B  00000033 Listen
  Local Socket:   ::..992
  Foreign Socket: ::..0
TCPIPB   00002396 UDP
  Local Socket:   ::..3507
  Foreign Socket: *..*
```

9.6 REXEC UNIX client command

This section provides an overview of the REXEC UNIX client command. The following topics are discussed:

- Description of the REXEC UNIX client command
- Configuration of the REXEC UNIX client command
- Verification of the REXEC UNIX client command

9.6.1 Description of the REXEC UNIX client command

The client can use the z/OS UNIX shell to enter the REXEC UNIX form of the rexec command. The command synonym, **orexec**, can be used.

Dependencies of the REXEC UNIX client command

To use REXEC, a REXEC daemon must be running on the remote host. The REXEC client passes the user name, password, and command to the remote REXEC daemon. The daemon provides automatic logon and user authentication, depending on the parameters that you set. The user ID specified on the REXEC command must be a valid user ID on the remote targeted system.

Considerations for using the REXEC UNIX client command

The UNIX REXEC client command does not provide the option of omitting the user ID and password. It cannot take advantage of the NETRC data set.

9.6.2 Configuration of the REXEC UNIX client command

Use the z/OS UNIX REXEC command (**rexec**/**orexec**) to execute a command on a remote host and receive the results on the local system, as shown in Example 9-36. The **rexec** command is a synonym for the **orexec** command in the z/OS UNIX shell. They both have the same format.

Example 9-36 REXEC UNIX client command format

```
orexec ? -d -l userid -p password -s portnum -C -V foreign_host command
```

The parameters **-d**, **-l**, **-p**, and **-s** are case sensitive and must be entered in lowercase letters. The user ID and password values might be case sensitive, depending on the remote operating system requirements.

Use a question mark (?) to obtain command help, as shown in Example 9-37.

Example 9-37 REXEC UNIX client command help

```
CS01 @ SC31:/u/cs01>rexec
Usage: orexec|rexec -V -d -l <user> -p <pwd>
                -s <port> fhost command
       options: -
                -?        display this message
                -d        turn on debug tracing
                -l <usr>  specifies remote login id
                -p <pwd>  specifies remote password
                -s <port> specifies server port
                -C        Uppercase messages
                -V        display APAR level
                -e <wait> select time limit

Example: orexec -d -l guest -p guest hostname ls -l
```

Available parameters include:

- Use **-d** to activate debug tracing.
- Use **-l** to specify the user ID to be used by the remote host.
- Use **-p** to specify the password for the user ID on the remote host.
- Use **-s** to specify the TCP port number of the rexec server on the remote host.
- Use -C to force messages to be displayed in uppercase characters.
- Use -V display the z/OS Communications Server version and release.
- Use *foreign_host* to specify the remote host name or IP address to which you are sending the indicated command.
- Use *command* to specify the command that is sent to the remote host. The command is composed of one or more words. The command you specify must not require a response from you to complete. The **rexec**/**orexec** command cannot interact with you after you enter data in the command format.

  The port number specified in the **rexec**/**orexec** command must match the port number on which the remote rexec server is listening. The default, if not specified here, is the port number specified on the *exec* entry within the /etc/services file, as shown in Example 9-38.

Example 9-38 Sample exec entry in /etc/services

```
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
```

z/OS REXEC UNIX requests can be submitted from the z/OS UNIX shell command line. Use the **rexec**/**orexec** command with user ID and password specified, as shown in Example 9-39.

Example 9-39 REXEC UNIC client command with user ID and password

```
READY
orexec -l CSO7 -p CSO7PSWD -s 512 10.1.1.20 netstat home
```

9.6.3 Verification of the REXEC UNIX client command

You can verify the results of the REXEC UNIX client command by reviewing the response you get back, as shown in Example 9-40.

Example 9-40 Results of REXEC UNIX client command

```
CSO1 @ SC31:/u/csO1>rexec -l CSO7 -p CSO7PSWD -s 512 10.1.1.20 netstat home
Home address list:
LinkName:   IUTIQDF4L
  Address:  10.1.4.21
    Flags:
LinkName:   IUTIQDF5L
  Address:  10.1.5.21
    Flags:
LinkName:   IUTIQDF6L
  Address:  10.1.6.21
    Flags:
LinkName:   VIPA1L
  Address:  10.1.1.20
    Flags:  Primary
LinkName:   OSA2080I
  Address:  10.1.2.21
    Flags:
LinkName:   OSA20A0I
  Address:  10.1.2.22
    Flags:
LinkName:   OSA20C0I
  Address:  10.1.3.21
    Flags:
LinkName:   OSA20E0I
  Address:  10.1.3.22
    Flags:
...
```

9.7 Problem determination for z/OS remote execution facilities

This section includes problem determination techniques and useful functions for the remote execution servers and client commands in the z/OS TSO and z/OS UNIX environments. We discuss the following topics in this section:

- Problem determination for TSO remote execution
- Problem determination for REXEC TSO with user ID and password
- Problem determination of REXEC TSO using NETRC
- Problem determination for the REXEC UNIX client command
- Recovery for server job table full condition
- Diagnostic messages for debugging

9.7.1 Problem determination for TSO remote execution

You can use the MVS MODIFY command to change some of the RXSERVE server operating parameters during execution. The following parameters cannot be changed while RXSERVE is executing:

- IPV6
- SECLABEL
- MAXCONN
- PREFIX

When the MODIFY (F) command is used to modify the RXSERVE parameters, RXSERVE responds with messages similar to those shown in Example 9-41.

Example 9-41 Modifying RXSERVE parameters dynamically

```
F RXSERVE,TSOPROC=$RISC,MSGCLASS=A,TSCLASS=Z,TRACE=LOG,EXIT=NOEXIT
EZA4400I Trace options: LOG,NOSEND,ALLCLIENTS
EZA4404I Parameters: MSGCLASS=A,TSCLASS=Z,TSOPROC=$RISC,MAXCONN=512,NOEXIT
```

9.7.2 Problem determination for REXEC TSO with user ID and password

When issuing a command to be executed on the remote host, do not place the command in quotation marks. The REXEC TSO client program cannot process the command stream properly if quotes are included.

Submitting a long-running command can cause the REXEC program to end abnormally with a 522 system abend code. You can avoid this issue by specifying TIME=1440 on the EXEC JCL statement. Job step timing is suppressed.

If the command to be executed on the remote host contains a slash (/), you must use a preceding slash (/) in the PARM= string, as shown in Example 9-42.

Example 9-42 Executing a remote command having an imbedded slash (/): PARM=

```
//CSO7C    JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CSO7
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='/-l oper6 -p cold2day 10.1.9.22 ls ./lpp/samples/*'
//SYSPRINT DD   DSN=CSO7.LPP.SAMPLES.C,DISP=SHR
```

A condition code of 12 will be set by the system when an REXEC batch request encounters any of the following error conditions:

- The client program cannot connect to TCP/IP.
- The host name cannot be resolved.
- The translation table cannot be found or loaded.

The REXEC client command can fail if any of the following error conditions occur:

- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP might not allow the REXEC client program access to the stack.
  - REXEC might not be configured or might be configured incorrectly.

- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- There is no REXEC server listening on the remote server.
  - You could be using an incorrect port number.
  - You could be using an incorrect server name or IP address.
- The remote system rejects the connect or submitted command.
  - The security might not be correctly set up for your user ID to log on to the remote server.
  - The security might not be correctly set up for your user ID to execute the requested command on the server.

9.7.3 Problem determination of REXEC TSO using NETRC

When issuing a command to be executed on the remote host, do not place the command in quotation marks. The REXEC TSO client program cannot process the command stream properly.

Submitting a long-running command can cause the REXEC program to end abnormally with a 522 system abend code. You can avoid this issue by specifying TIME=1440 on the EXEC JCL statement. Job step timing is suppressed.

If the command to be executed on the remote host contains a slash (/), you must use a preceding slash (/) in the PARM= string, as shown in Example 9-43.

Example 9-43 Executing a remote command having an imbedded slash (/)

```
//CS07E JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS07
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='/-l oper6 -p cold2day 10.1.9.22 ls ./lpp/samples/*'
//SYSPRINT DD   DSN=CS07.LPP.SAMPLES.E,DISP=SHR
```

The REXEC client command can fail if any of the following error conditions occur:

- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP might not allow the REXEC client program access to the stack.
  - REXEC might not be configured or might be configured incorrectly.

- There is no REXEC server listening on the remote server.
  - You could be using an incorrect port number.
  - You could be using an incorrect server name or IP address.
- The remote system rejects the connect or submitted command.
  - The security might not be correctly set up for your user ID to log on to the remote server.
  - The security might not be correctly set up for your user ID to execute the requested command on the server.

9.7.4 Problem determination for the REXEC UNIX client command

The REXEC UNIX client command can fail if any of the following error conditions occur:

- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP might not allow the REXEC client program access to the stack.
  - REXEC might not be configured or might be configured incorrectly.
- There is no REXEC server listening on the remote server.
  - The port number might be incorrect.
  - The server name or IP address might be incorrect.
- The remote system rejects the connect or submitted command.
  - The security might not be correctly set up for your user ID to log on to the remote server.
  - The security might not be correctly set up for your user ID to execute the requested command on the server.

9.7.5 Recovery for server job table full condition

If the REXECD server is started with the PURGE=N option, then the entries in the table remain until the server is recycled, even if the jobs on the JES queue are purged by the user. If you cannot take any recovery action when the server cannot assign any new job numbers, then:

- Remote jobs are lost.
- Several jobs fail before the condition is noted.
- Message is issued currently only to the server's JES spool.

When running a server 24x365, this condition can occur eventually if the server was started with the PURGE=N option. It can also occur when using a shared JES spool with servers running on other systems and using the same prefix.

The REXECD server, now, attempts to clean up table entries for jobs that are purged from the JES spool when starting the server with the PURGE=N option.

Additionally, messages EZA4434I or EZA4435E are written to the console when the server detects table full:

- EZA4434I is issued when job table is 85% full:

  ```
  EZA4434I rexecd: Number of available job numbers is being depleted
  ```

- EZA4435E is issued if the REXECD server is not recycled after EZA4434I and if the job table becomes full:

  ```
  EZA4435E rexecd: Number of available jobs is depleted
  ```

- The REXECD server tries to find candidate job numbers for reuse based on the time stamps of internal job table entries.

When messages EZA4434I or EZA4435E display, you have the following options:

- Change the setting of the PURGE operand to PURGE=Y so that job numbers are purged immediately when no longer in use.
- Delete jobs from the JES spool.
- Restart the server.

By updating your automation to identify the EZA4434I and EZA4435E messages, you can use automation to recycle the server and bring the situation to the attention of the console operator, who can manually recycle the server.

9.7.6 Diagnostic messages for debugging

In an environment that has heavy REXEC server demands, it is often difficult to correlate the failing request with the provided diagnostic trace entries. This issue is compounded when a business unit is running a server 24x365.

The REXECD server provides diagnostic messages that it is easy to correlate with a single remote job request and to identify failing requests.

Diagnostic messages include:

```
EZA4440I Closing connection with remote_session from local_session
EZA4436I SSCSARAY[0] jobnumber 80 ACTIVE
EZA4437I SSCSARAY[0] jobnumber 40 WAITING
EZA4438I SSCSARAY[0] jobnumber 20 COMPLETED
EZA4439I SSCSARAY[0] jobnumber 10 HELD
EZA4442I SSORT(next): 00000004 NO MORE OUTPUT ON THE JES QUEUE
```

Trace messages have a standard format. Each message is prefaced by the socket number that is associated with the message. After the socket number, the message ID value is displayed, followed by the message text.

Table 9-1 on page 360 lists the trace message IDs and the associated message text.

Table 9-1 Example trace to the JES spool file of the server

1	socket 0: EZA4381I Accept 2 from *XX.XX.XX.XX* on 512
2	socket 2: EZA4382I Connecting on 3 to *XX.XX.XX.XX*:1255 socket 2: EZA4436I SSCSARAY[0] JOB00048 80 ACTIVE socket 0: EZA4381I Accept 4 from *XX.XX.XX.XX* on 514 socket 2: EZA4436I SSCSARAY[0] JOB00048 80 ACTIVE socket 2: EZA4436I SSCSARAY[0] JOB00048 80 ACTIVE socket 4: EZA4382I Connecting on 5 to *XX.XX.XX.XX*:1584 socket 2: EZA4438I SSCSARAY[0] JOB00048 20 COMPLETED socket 2: EZA4385I SSSORT(CTRL): 00000000 socket 2: EZA4392I S99ret: 00000000, T RSHD USER35.RSHD2.JOB00048.D0000102.? socket 2: EZA4393I S99ret: 00000000
3	socket 2: EZA4442I SSSORT(next): 00000004 NO MORE OUTPUT ON JES SPOOL
4	socket 4: EZA4440I Closing connection with 4 from *XX.XX.XX.XX*
5	socket 5: EZA4442I Closing socket 5 to *XX.XX.XX.XX*:1584

The first token in the trace line identifies the socket to which the trace entry applies. Socket 0 is the listening socket. The EZA4381I message identifies the socket on which the request is processed and the IP address and port on which that the request was received. Port 512 is for REXEC requests and port 514 is for RSH requests.

In this table:

1. Indicates an REXEC request was received on socket 2 for the specified IP address. Subsequent entries beginning with "`socket 2:`" indicate activity that is occurring during the processing of this request. Message EZA441I is issued when this socket is closed.
2. Shows EZA4382I, which identifies the socket (3) and the IP address and port that the server uses to connect to the client. This port (1255) was in the packet that the client sent to the server. Message EZA4442I is issued when this socket is closed. Typically, you see only the EZA4382I and EZA4442I messages for this socket.
3. Shows the return code from the dynamic allocation of the JES data set back to the client.
4. Indicates that the request is completed and that the socket is closed.
5. Indicates that the error socket is closed.

9.8 Additional information sources for remote execution and remote shell

See the following sources for information about remote execution servers and protocols:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS UNIX System Services Planning*, GA22-7800

Tip: For descriptions of security considerations that affect specific servers or components, see "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

10

Domain Name System

The Domain Name System (DNS) is a client-server solution in which *name servers* provide information about host systems and their IP addresses. This chapter describes the z/OS DNS, which uses the Berkeley Internet Name Domain (BIND) software, the accepted standard of DNS. BIND was developed at the University of California, Berkeley, and is currently maintained by the Internet Software Consortium (ISC). This chapter focuses on the DNS BIND 9 functions that are available in the z/OS Communications Server.

Note: z/OS V1R13 is planned to be the last release in which the BIND 9.2.0 function will be available. The caching only server function can be replaced using the resolver caching function described in *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996.

Primary or secondary authoritative server function can be moved to BIND on Linux for System z or BIND on an IBM blade in an IBM zEnterprise™ BladeCenter® Extension (zBX).

This chapter discusses the following topics.

Section	Topic
10.1, “Conceptual overview of the DNS name server” on page 362	The basic concepts of DNS.
10.2, “Authoritative DNS server” on page 365	Key characteristics of an authoritative DNS server.
10.3, “Caching-only DNS server” on page 367	Key characteristics of a caching-only DNS server on z/OS, and how to configure and use it.
10.4, “Automated domain name registration” on page 379	How to configure and use ADNR facilities.
10.5, “Problem determination for DNS service” on page 389	Problem determination techniques for the DNS servers: the caching-only DNS and the ADNR.
10.6, “Additional information sources for DNS” on page 394	References to additional reading sources for the z/OS DNS server.

10.1 Conceptual overview of the DNS name server

As illustrated in Figure 10-1, the DNS name server is one of the standard applications provided with the z/OS Communications Server. The DNS server uses the z/OS UNIX Sockets interface to send and receive its packets through the Logical and Physical File Systems.

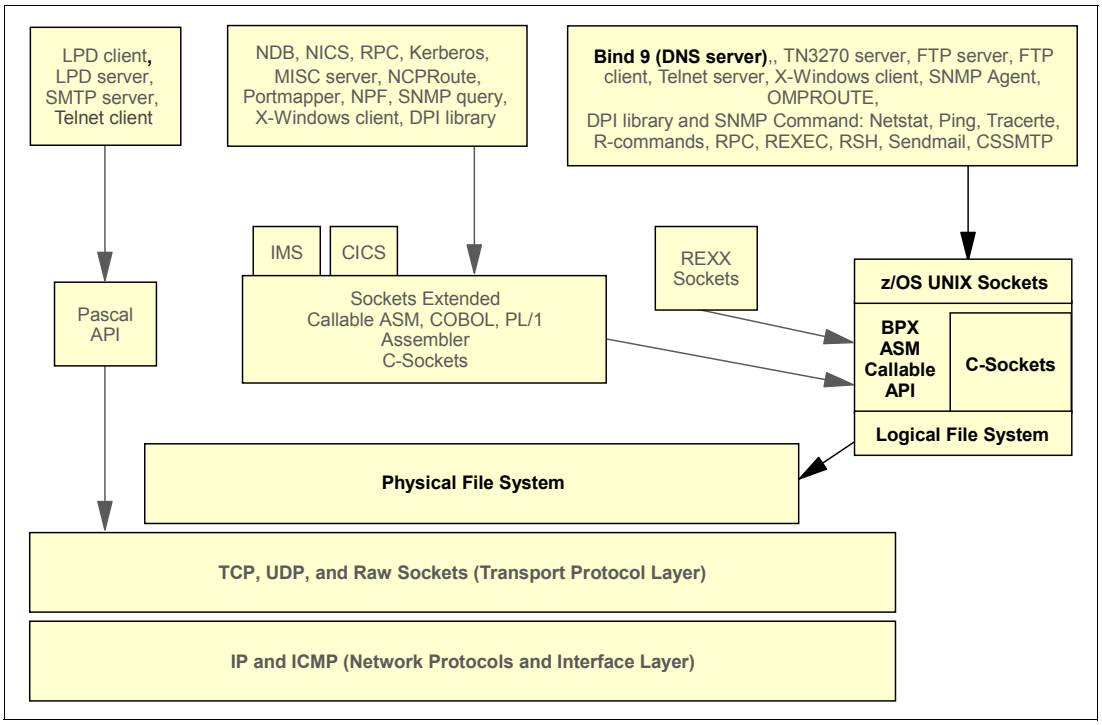


Figure 10-1 DNS service

Name servers provide names and IP addresses to clients called *resolvers*. The information helps to cross-reference an IP address to a name, or the name to an IP address.

This section discusses the following concepts:

- What is Domain Name System
- How does Domain Name System work
- How can Domain Name System be applied
- Considerations about z/OS DNS BIND 9 implementation

10.1.1 What is Domain Name System

DNS was originally developed to make things easier for human users: to enable the use of symbolic names rather than the actual, numeric, 32-bit IPv4 or 128-bit IPv6 addresses upon which TCP/IP communications ultimately depends. The use of symbolic names supported by DNS, however, has turned out to be far more important than just a convenience because it provides separation of applications from their physical infrastructure. For example, without the use of symbolic names, if you needed to change the IP address of your TN3270E Telnet server (perhaps because you are moving the server) you would have to change the (potentially) thousands of clients configured to use its original IP address. With symbolic names, you need only change the DNS entry that corresponds to the TN3270E Telnet server name and the clients will learn of the changed IP address when they use its name.

DNS also provides IP address-to-host name mapping. The DNS defines a special domain called *in-addr.arpa* to translate IPv4 addresses to host names, and the *ip6.int* and *ip6.arpa* domains for IPv6 address-to-host name translation. This kind of mapping is useful for producing output (host names) that is easy to read.

10.1.2 How does Domain Name System work

The relationships between resolvers (the clients) and DNS servers are illustrated in Figure 10-2. The authoritative servers are the Primary and Secondary servers with IP addresses of a.b.c.d and e.f.g.h. The DNS used with ADNR is an authoritative server.

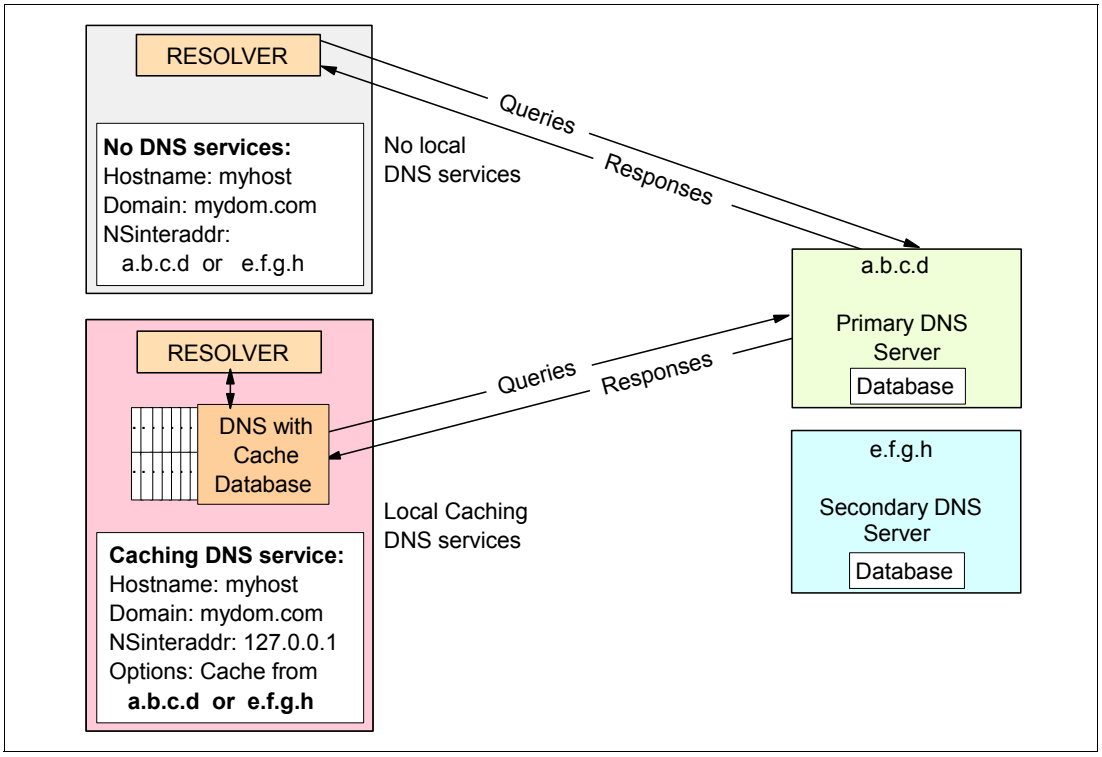


Figure 10-2 Caching DNS server and the resolver

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. 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 periods. Each label represents an increasingly higher domain level within a network. The complete name of a host connected to a large network generally has more than one subdomain, as shown in the following examples:

```
wtsc30.itso.ibm.com
wtsc31.itso.ibm.com
itsodns.itso.ibm.com
```

A domain name server requires the FQDN. The client resolver combines the host name with the domain name to create the FQDN before sending the name resolution request to the domain name server.

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. For DNS performance or availability reasons, you might want to partition your name space into zones and make each DNS server the *primary* for part of the name space and the *subordinate* (backup) for another part.

10.1.3 How can Domain Name System be applied

DNS can be implemented on the z/OS platform as one or more types of DNS servers. These types include authoritative, caching-only, forwarders, and stealth. A single server can perform multiple functions. For example, an authoritative DNS server can provide caching for names from other zones.

In this book we discuss these types of DNS services:

- Authoritative DNS server
- Caching-only DNS server
- Automated Domain Name Registration (ADNR)

Having your own authoritative DNS server gives you administrative control over your own space of names. Therefore, if you need a new name, you can simply add it to your server database rather than having to ask someone else to allocate a name for you. It also puts you in control of the availability and scalability of your own DNS service—not having to depend upon the DNS service provided by another organization.

You can use caching servers to create a 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. Also, there is no clerical maintenance to be performed with a caching-only server.

Note: Configure your resolver so that it will check with the local caching-only DNS server first. However, if that server is unavailable for any reason, your resolver should also be able to direct queries to an external DNS (by using multiple NSINTERADDR statements in TCPIP.DATA. This setup is shown in Example 10-16 on page 377.

Most organizations have little need to design, implement, manage, and administer their own name space and authoritative DNS services for their IBM System z environment. Rather, that role is usually handled more globally for the organization and is usually implemented on external (non-z/OS) platforms. However, a caching-only DNS server on the mainframe can play an important role, especially to improve the performance of z/OS web servers and IBM WebSphere® applications. Caching DNS information about board can reduce CPU utilization and increase responsiveness to the numerous web transactions that require DNS services.

Note: BIND 4.9.3 is no longer supported in z/OS Communications Server - TCP/IP component.

We therefore focus our implementation on setting up and using a BIND 9 Caching-only DNS on the z/OS system.

Before we show the implementation steps, it is important to list some considerations about the z/OS BIND 9 history and implementation issues, as follows:

- BIND 9 allows IPv6-type records in zone data and better transaction security among servers and clients, and zone data authentication capability. It also has the **rndc** utility to replace and complement UNIX signals for name server local and remote control.
- The BIND 9.2 name server makes DNS server-to-server and client-to-server IPv6 connections possible, adding name server configuration options for IPv6 connections and tuning. BIND 9.2 also provides the **rndc** utility with a larger set of commands than was available with BIND 9.1. BIND 9.2 **rndc** is not compatible with BIND 9.1 **rndc**.
- The BIND 9 *nsupdate* utility enables client hosts and many DHCP servers to dynamically and securely register their name and address mappings. The z/OS BIND 9 name server is generally compatible with network DHCP servers.
- In a future release of z/OS, the BIND 9.2.0 function will be removed from z/OS Communications Server. Customers who currently use or plan to use the z/OS BIND 9.2.0 function as a caching-only name server must use the Resolver function to cache DNS responses after the BIND 9.2.0 function is removed. For more information, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996. Customers who currently use or plan to use the z/OS BIND 9.2.0 function as a primary or secondary authoritative name server should consider using BIND on Linux for System z.

Note: There are a number of considerations to be aware of when running the BIND 9 server in a multiple stack environment (CINET). See the BIND 9 advanced topics section in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details about these restrictions.

10.1.4 Considerations about z/OS DNS BIND 9 implementation

When BIND v9 DNS server is started from either a start procedure or from the z/OS UNIX shell command line, the assigned user ID by default is defined as a superuser [UID(0)]. If the customer wants, z/OS Communications Server permits DNS Bind 9 server to run with a non zero UID. To be able to do that, the user ID associated with the BIND 9 name server must have permitted access to the BPX.SUPERUSER class profile.

10.2 Authoritative DNS server

An authoritative DNS server is responsible for one or more zones, managing the names and IP address associations within that zone. For reasons stated in 10.1.3, “How can Domain Name System be applied” on page 364, we provide only a description of the authoritative DNS server, and do not show its configuration or implementation.

10.2.1 Description of an authoritative DNS server

Authoritative DNS servers are designated network nodes that maintain a database of information about all nodes in some part (zone) of the domain name space. See Figure 10-2 on page 363 to see the role of the authoritative server contrasted with that of the caching-only server.

If a domain name server receives a query about a host for which it has information in its database or cache, it returns all the address records associated with the host to the client (some hosts might have more than one IP address). If the domain name server does not have the requested information, it can query other name servers for it. This process is called iterative resolution. The local name server successively queries other name servers, each of which responds by referring to a name server that is closer to the authoritative name server for the target domain. Ultimately, 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.

Updates and maintenance changes are reflected in the master name server. The subordinate name servers update their databases by contacting the master name server at regular intervals or possibly (BIND 9) after being notified of an update by the master name server.

There are two types of authoritative servers: master (primary) and subordinate (secondary). Each zone must have only one master name server, and it should have at least one subordinate name server for backup to eliminate the single point of failure for this important service. Any given name server can take on either role, as defined by its configuration file.

Dependencies of an authoritative DNS server

If you set up your own authoritative DNS server you will, by definition, be required to manage and administer the space of names in your zone. In addition, given that yours is the *final authority* about the names in your zone, you must be careful to implement sufficient availability and scalability for your server. If your authoritative DNS service becomes unavailable (because of overload or outage), even if the physical network infrastructure remains in tact, users will perceive an outage for any application accessed by name in your zone.

Note: If you use a VIPA address in the DNS for zone delegation, make sure that VIPA address is the SOURCEVIPA address on that stack. Otherwise, certain name server queries could fail. For further information, see 2.5.5.10 in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Considerations for using an authoritative DNS server

While having your own authoritative DNS server gives you a great deal of control, it comes with significant *responsibility*:

- As mentioned, you need to design your DNS service for sufficient scalability and availability to meet your business requirements. Unavailability of DNS service would likely result in significant application *outage*, as perceived by users.
- In addition, you need to provide ongoing management and administration of your DNS servers.
- Finally, providing DNS service could require significant processor resources, especially if you implementation requires security (DNSSEC, authenticating DNS data with digital signatures) implementation.

10.3 Caching-only DNS server

All name servers cache (store) the data they receive in response to a query. As the name suggests, caching-only DNS servers have no zone responsibility and must use other name servers to resolve name requests that they receive. They do, however, cache the resulting information and use it to answer future queries.

Note: When planning to set up caching-only DNS servers in your environment, check your current Resolver configuration and make sure the host names you plan to resolve by the caching-only DNS servers are not being resolved by the Resolver. The search order for the base Resolver configuration files is as follows:

1. GLOBALTCPIPDATA
2. RESOLVER_CONFIG environment variable
3. /etc/resolv.conf
4. SYSTCP DD-name
5. userid.TCPIP.DATA
6. jobname.TCPIP.DATA
7. SYS1.TCPPARMS(TCPDATA)
8. DEFAULTTCPIPDATA
9. TCPIP.TCPIP.DATA

In this section, we describe the following topics:

- Description of a caching-only DNS server
- Configuration of a caching-only DNS server
- Activation and verification of a caching-only DNS server

Note: To provide better system performance, consider eliminating redundant network flows to DNS servers by exploiting the Resolver DNS Cache. This feature uses the resolver for system-wide caching of Domain Name System (DNS) responses. Resolver DNS cache provides the same function as a caching DNS Server with better system performance. For details about the Resolver DNS Cache, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996.

10.3.1 Description of a caching-only DNS server

A caching-only server 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 the root name servers are acquired from the servers listed in the hints file and the name and file path of the hints file are specified in the name server's configuration file. See Figure 10-2 on page 363 to see the role of the authoritative server contrasted with that of the caching-only server.

Dependencies of a caching-only DNS server

Because the caching-only DNS server has no domain name database of its own to manage, it is completely dependent upon other DNS servers.

If you want to use an application-specific Dynamic VIPA (DVIPA) as the address of the name server, then the name server must BIND to the well-known port for DNS (UDP port 53) on the DVIPA.

Considerations for using a caching-only DNS server

All cached information is lost if a caching-only DNS server is restarted. Also, by definition, a caching-only DNS server gives your no control over your name space. Rather, you will need to work with your local naming authority (the person or people who control the authoritative name server for your zone).

Note: Use z/OS UNIX or TSO nslookup with each NSINTERADDR used in TCPIP.DATA to ensure you receive the expected results. Some name server clients on other platforms might 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.

10.3.2 Configuration of a caching-only DNS server

The name resolution process is an example of a client/server relationship in which clients, through their resolvers, request name resolution service from a name server.

z/OS Communications Server: IP Configuration Guide, SC31-8775, contains a comprehensive discussion on setting up the BIND 9 Caching-only server. See that manual for details. A summarized checklist is provided here to assist with understanding the above reference.

The tasks for configuring a caching-only server are discussed in the following sections:

- Create a BIND 9 configuration file for the server.
- Create an environment variables file for the server.
- Update the TCP/IP stack profile port reservation list for the server.
- Create the name server start procedure.
- Create the hints file (root server).
- Create the loopback file.
- Configure logging options for the server.
- Implement the syslog daemon (syslogd) or alternative log file.
- Determine the run mode of the server: swappable or nonswappable.
- Change server user ID to run with BPX.SUPERUSER authorization.

Create a BIND 9 configuration file for the server

We created a working directory (`dnsdatb31`) in the z/OS UNIX file under the main `/etc` directory where all of our DNS server files could be placed. We explicitly defined some of the files. The others were created dynamically by the server task. We coded a directory option statement pointing to the desired directory.

A directory list of /etc/dnsdatb31 resulted in the display shown in Example 10-1.

Example 10-1 Listing of the directory for the NAMED9B server, /etc/dnsdatb31

```
Directory List
EUID=0   /SC31/etc/dnsdatb31/
  Type  Perm  Changed-EST5EDT   Owner      Filename
_ Dir    777  2010-09-26 19:10  CSO7       .
_ Dir    755  2010-09-30 17:02             ..
_ File   777  2010-10-02 00:17  SYSPROG    db.cache.v9
_ File   777  2010-10-02 00:18  CSO7       db.loopback.v9
_ File   666  2010-09-26 19:10  SYSPROG    dns.log
_ File   777  2010-10-02 00:14  SYSPROG    named9b.conf
_ File   777  2010-10-02 00:16  CSO7       named9b.env
```

Example 10-2 is the configuration file for a caching-only server.

Example 10-2 Configuration file for the NAMED9B task

```
BROWSE -- /SC31/etc/dnsdatb31/named9b.conf --------- Line 00000000 Col 001 050
#  named.conf file for named9b for sc31

options {
   directory "/etc/dnsdatb31";
   pid-file "/etc/dnsdatb31/named9b.pid";
   statistics-file "/etc/dnsdatb31/named9b.stats";
   forward only;
   forwarders {10.12.6.7; 10.12.12.7; 10.12.4.7;};
   max-ncache-ttl 1800;
   max-cache-ttl 36000;
};

logging {
   channel "myfile" {
      file "/etc/dnsdatb31/dns.log"
       versions 99
       size 30m ;
      severity info ;
      };
   category "default" { "myfile"; };
   category "queries" { "myfile"; };
   category "general" { "myfile"; };
   category "config"  { "myfile"; };

zone "0.0.127.in-addr.arpa" in {
   type master;
   file "/etc/dnsdatb31/db.loopback.v9";
};

zone "." in {
   type hint;
   file "/etc/dnsdatb31/db.cache.v9";
};
```

A sample configuration file for a caching server is located in `/usr/lpp/tcpip/samples/caching.conf`, and can be copied to the file that you want to use for your configuration file. Because we chose to put all of our files in `/etc/dnsdatb31`, we copied the sample file to `/etc/dnsdatb30/named9b.conf`, and pointed our started task proc JCL to it by using the **-c** parameter.

Create an environment variables file for the server

We created an environment variables file to specify stack affinity, resolver config, job name, and DNS version. The BIND 9 server is a generic server and binds to all interfaces on all stacks in a CINET environment. If you want it to have stack affinity, you can specify stack affinity in the environment variables file like we did. The server can be configured to listen on a subset of available interface addresses if desired. It is not necessary to specify the job name, because the system suffixes the started task name with the number one (1) when BIND 9 starts. Our started task was NAMED9B, but specified a job name of NAMEDB, and the forked job resulted in a name of NAMEDB1. If you want the job name to be different from the started task name, you can set the job name in the environment variables file. The STDENV file that we used is shown in Example 10-3.

Example 10-3 Sample STDENV file

```
BROWSE -- /SC31/etc/dnsdatb31/named9b.env ---------- Line 00000000 Col 001 045
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
_BPX_JOBNAME=NAMEDB
DNS_VERSION=v9
```

Note: You can use the *_CEE_ENVFILE* environment variable in the PARM field of the JCL to point to a file that contains other environment variables. The file can be a UNIX file, a zFS, or a z/OS MVS data set. When it is an MVS data set, the data set must be allocated with RECFM=V.

RECFM=F must not be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in z/OS UNIX. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Update the TCP/IP stack profile port reservation list for the server

The PORT reservation list would look like Example 10-4.

Example 10-4 Port reservation for the DNS server

```
PORT
   53 TCP NAMEDB1
   53 UDP NAMEDB1
```

The PORT reservation for port 53 can specify the started task name as the job name (suffixed with the number one (1)), or the UNIX shell address space name can be specified. Because it is less clerical effort, you can use OMVS as seen in Example 10-5. Using this definition would avoid any name conflicts in the future, especially if you change the name of the started task.

Example 10-5 Port reservation for the DNS server using OMVS

```
PORT
   53 TCP OMVS
   53 UDP OMVS
```

Create the name server start procedure

A sample procedure for the BIND 9 server is located in SEZAINST(NAMED9). We modified the sample to specify our own configuration file and environment variable file. The start procedure JCL for the BIND 9 DNS server is stored in a system proclib. Make sure you use a valid PARM string acceptable to BPXBATCH. It requires that the PARM string field be a contiguous string to column 71, with a continuation character in column72, then the remainder of the parm field continuing on the next line in column 16. Also notice the specification of the configuration file in the PARM field, using the **-c** parameter. The environment variables file is pointed to by using the //STDENV DD statement. BPXBATCH automatically detects the presence of the //STDENV DD statement, and you do not have to use the _CEE_ENVFILE environment variable to point to it. The debug trace level can be specified when starting the procedure. The default debug level is set to zero. Additional parameters can be included in the PARMS= specification at startup, as seen in Example 10-6.

Example 10-6 JCL procedure for the BIND 9 DNS server

```
BROWSE    SYS1.PROCLIB(NAMED9B) - 01.27             Line 00000000 Col 001 080
//NAMED9B  PROC PARMS='-d 0'
//NAMED9B  EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
//         PARM=('PGM /usr/lpp/tcpip/sbin/named -V v9 -c
//             /etc/dnsdatb&SYSCLONE./named9b.conf &PARMS')
//STDENV   DD PATH='/etc/dnsdatb&SYSCLONE./named9b.env',
//             PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Create the hints file (root server)

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. If your BIND 9 server is going to run on a system that is behind a firewall, then your hints file will point to one or more of your normal DNS servers that can resolve intranet and Internet names. However, if your system is not behind a firewall and has access to the Internet DNS root servers, then you must install the hints file provided by the Internic. To obtain a hints file, point your web browser to the following website and retrieve the file named.root from the domain subdirectory:

`ftp://ftp.rs.internic.net`

Because we are defining a caching-only server, we chose to use forward-only, specifying one or more of our own internal DNS servers, and leave the hints file (/etc/dnsdatb31/db.cache) empty.

Note: Before you specify the remote DNS servers IP addresses in *forwarders* statement with *forward only*, make sure your Caching-only server can access them, otherwise the local caching-only server could not resolve the host names.

Create the loopback file

The sample loopback file for BIND 9 is located in /usr/lpp/tcpip/samples/db.loopback.v9, and can be copied to the file you want to use for your loopback file. Because we placed all of our BIND 9 DNS related files into /etc/dnsdatb31, we copied the sample file to /etc/dnsdatb31/db.loopback.v9 and customized it to reflect our domain name. For a comprehensive discussion on the loopback file for caching servers, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Example 10-7 is the loopback file.

Example 10-7 Loopback file for BIND 9 DNS server

```
BROWSE -- /SC31/etc/dnsdatb31/db.loopback.v9 ------- Line 00000000 Col 001 065
;
;   /etc/dnsdata/db.loopback.v9
;
; Default TTL value, $TTL is for BAND 9
$TTL 86400
0.0.127.in-addr.arpa. IN SOA itsodns.itso.ibm.com. itsodns_adm.itso.ibm.com. ( 1
     1
     10800
     3600
     604800
     86400   )
0.0.127.in-addr.arpa.   IN   NS  itsodns.itso.ibm.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
```

In this example, the loopback file is a z/OS UNIX file, and the record length has no 72 character limitation on one line, as shown at **1**. The SOA statement should be followed by the authoritative_name_server, resp_person and the first parenthesis in the same line. Do not omit either of these, or you will receive the error messages, shown in Example 10-8, in the UNIX log file of the name server (/etc/dnsdatb31/dns.log) when it starts and loads the loopback file.

Example 10-8 Error messages for loopback file

```
EZZ9190I dns_rdata_fromtext: /etc/dnsdatb31/db.loopback.v9:17: near eol:
unexpected end of input
EZZ9688I zone 0.0.127.in-addr.arpa/IN: loading master file
/etc/dnsdatb31/db.loopback.v9: unexpected end of input
```

For the statement syntax in loopback file, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Configure logging options for the server

A wide variety of logging options for the name server can be configured using the *logging* statement. Its *channel* phrase associates output methods, format options, and severity levels with a name that can then be used with the category phrase to select how various classes of messages are logged. Only one logging statement is used to define as many channels and categories as are wanted. For details of the logging statement, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for a discussion about how to use logging. For details about the syntax and format of the logging statement, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Example 10-9 shows the statements we used. Our statements are not typical, but simply show the types of logging channels and categories you can create and manage.

Example 10-9 Sample logging options

```
logging {
   channel "myfile" {
      file "/etc/dnsdatb31/dns.log"
       versions 99
       size 30m ;
      severity info ;
      };
   category "default" { "myfile"; };
   category "queries" { "myfile"; };
   category "general" { "myfile"; };
   category "config"  { "myfile"; };
};
```

Implement the syslog daemon (syslogd) or alternative log file

For BIND 9 only, unless syslogd is running, no messages will be produced by NAMED during name server initialization. This includes event logging and any syntax errors that might be detected in the configuration file. Not performing this step complicates problem determination, especially for failures at startup. For descriptions of the syslog file and the syslog daemon, see Chapter 1, “The syslog daemon” on page 1.

Determine the run mode of the server: swappable or nonswappable

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
```

If you want the name server to run in a *nonswappable* state, use the following commands:

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
PERMIT BPX.STOR.SWAP CLASS(FACILITY) ID(userid) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

The *userid* in the above command is the user ID under which your BIND 9 started task is intended to run.

Change server user ID to run with BPX.SUPERUSER authorization

If required, change the user ID that is associated with BIND 9 from UID(0) to a unique nonzero UID and permit the user ID to access BPX.SUPERUSER class profile. Complete the following steps:

1. Define and activate the BPX.SUPERUSER resource in the FACILITY class:

   ```
   RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
   SETROPTS CLASSACT(FACILITY)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

2. Change the BIND 9 user ID from UID(0) to a unique nonzero UID using these commands:
   - Cleanup the UID definition:

     ```
     ALTUSER userid OMVS(NOUID)
     ```

– Make RACF assign the UID automatically:

```
ALTUSER userid OMVS(AUTOUID)
```

– Assign a specific nonzero UID, such as x:

```
ALTUSER userid OMVS(UID(x))
```

3. Permit the BIND 9 user ID to access BPX.SUPERUSER in read mode

```
PERMIT BPX.SUPERUSER
CLASS(FACILITY) ID(userid)
ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

10.3.3 Activation and verification of a caching-only DNS server

The startup of the server and its responsiveness can be verified as follows:

- Start the name server task
- Verify that the name server started correctly.
- Use netstat to verify that the server is listening on the intended port.
- Verify that the name server can accept queries.
- Browse the log to see the latest query activity just created.
- Stop the caching-only DNS server to verify it will shut down cleanly.

Start the name server task

The BIND 9 server can be started using the MVS START (S) command, the z/OS UNIX shell named command, or using automations. We started our NAMED9B procedure using the MVS START command:

```
S NAMED9B
```

Sample messages issued at startup are shown in Example 10-10. The original job spawns a new job called NAMEDB1, and then the original job terminates.

Example 10-10 NAMED9 startup messages

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 1 --  N O D E
S NAMED9B
$HASP100 NAMED9B  ON STCINRDR
IEF695I START NAMED9B  WITH JOBNAME NAMED9B  IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 NAMED9B  STARTED
$HASP395 NAMED9B  ENDED
+EZZ9095I STARTING NAMED, BIND 9.2.0
+EZZ9130I NAMED, BIND 9.2.0 IS RUNNING
```

After startup, the process ID of the NAMED9B task is placed into the PID file, as shown in Example 10-11.

Example 10-11 PID file containing the process ID of the NAMED9B task

```
BROWSE    /SC31/etc/dnsdatb31/named9b.pid           Line 00000000 Col 001 006
******************************** Top of Data ********************************
131204
****************************** Bottom of Data ******************************
```

The PID value can be used on a **kill** command to stop the task. If the NAMED9B task was started using the MVS START (S) command, the MVS STOP (P) command can be used to stop the task, as in Example 10-12.

Example 10-12 Stopping the NAMED9B task using the UNIX shell

```
kill -TERM 131204
or
kill -TERM $(cat /etc/dnsdatb31/named9b.pid)
or
P NAMED9B
```

Verify that the name server started correctly

Browse the log where you directed the server to log and verify the results of startup. As shown in Example 10-13.

Example 10-13 NAMED9 log immediately after startup

```
BROWSE    /SC31/etc/dnsdatb31/dns.log               Line 00000000 Col 001 080
******************************** Top of Data ********************************
EZZ9163I Found DNS_VERSION environment variable with value of v9
EZZ9171I LPAR mode detected. Using 2 CPUs for -n option
EZZ9547I starting named, BIND 9.2.0 -V v9 -c /etc/dnsdatb31/named9b.conf -d
EZZ9095I STARTING NAMED, BIND 9.2.0
EZZ9217I Running non-swappable
EZZ9540I using 2 CPUs
EZZ9126I loading configuration from '/etc/dnsdatb31/named9b.conf'
EZZ9573I set maximum stack size to 0: invalid file
EZZ9573I set maximum data size to 0: invalid file
EZZ9046I listening on IPv4 interface OSA2100LNK      , 9.12.4.213#53
EZZ9046I listening on IPv4 interface OSA2120LNK      , 9.12.4.214#53
EZZ9046I listening on IPv4 interface loopback127     , 127.0.0.1#53
EZZ9130I NAMED, BIND 9.2.0 IS RUNNING
****************************** Bottom of Data ******************************
```

There could be warning messages indicating that optional files were not found. Processing continues normally without them. However, if you intend to use **rndc** to manage your server, then you must create the rndc environment (see *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781).

Use netstat to verify that the server is listening on the intended port

The **netstat** commands shows current socket and UDP connections. Verify that the name server that you started is listening on the intended stacks, interfaces, and port. The resulting **netstat** display that we used is shown in Example 10-14.

Example 10-14 Verifying NAMED9B connections with NETSTAT -c (partial results)

```
CS02 @ SC31:/u/cs02>netstat -p tcpipb -c
MVS TCP/IP NETSTAT CS V1R12       TCPIP Name: TCPIPB          01:34:55
User Id  Conn     State
-------  ----     -----
...
NAMEDB2  00000626 Listen
  Local Socket:   10.1.1.20..53
  Foreign Socket: 0.0.0.0..0
```

```
NAMEDB2  0000061A Listen
  Local Socket:   10.1.2.21..53
  Foreign Socket: 0.0.0.0..0
NAMEDB2  0000061C Listen
  Local Socket:   10.1.2.22..53
  Foreign Socket: 0.0.0.0..0
NAMEDB2  0000061E Listen
  Local Socket:   10.1.3.21..53
  Foreign Socket: 0.0.0.0..0
.
NAMEDB2  00000634 Listen
  Local Socket:   10.1.8.30..53
  Foreign Socket: 0.0.0.0..0
NAMEDB2  00000632 Listen
  Local Socket:   10.1.8.10..53
  Foreign Socket: 0.0.0.0..0
...
```

Verify that the name server can accept queries

First, use the normal TSO *nslookup* process to verify that you can retrieve DNS information using the usual TCPDATA file that has the standard *external DNS server* specified on the NSINTERADDR/NAMESERVER statement. This method used file 'TCPIPB.TCPPARMS(*DATAB31*)' and is shown in the following example. It does not make use of the new caching-only server that we just started. Pointing to and using an external DNS is shown in Example 10-15.

Example 10-15 Uses external DNS server

```
BROWSE TCPIPB.TCPPARMS(DATAB31) - 01.05              Line 00000000 Col 001 080
TCPIPJOBNAME TCPIPB
HOSTNAME WTSC31
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  10.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

free ddn(systcpd)
alloc ddn(systcpd) dsn('TCPIPB.TCPPARMS(DATAB31)') shr
nslookup

 EZB3170I Default Server:  itsodns.itso.ibm.com
 EZB3172I Address:  10.12.6.7
 EZB3042I >
wtsc30
 EZB3170I Server:  itsodns.itso.ibm.com
 EZB3172I Address:  10.12.6.7
 EZB3170I Name:    wtsc30.ITSO.IBM.COM
 EZB3172I Address:  10.1.1.10

 EZB3042I >
wtsc31
```

```
 EZB3170I Server:  itsodns.itso.ibm.com
 EZB3172I Address:  10.12.6.7
 EZB3170I Name:    wtsc31.ITSO.IBM.COM
 EZB3172I Address:  10.1.1.20
 EZB3042I >
```

Next, use the TSO nslookup process to verify that you can retrieve DNS information using the special TCPDATA file that has been set up with the NSINTERADDR/NAMESERVER statement pointing to the *loopback address* (127.0.0.1) instead of pointing to the remote name server directly. An example of this method uses file 'TCPIPB.TCPPARMS(*DATAB31N*). It indicates to the resolver to use our own DNS server that is listening on port 53 on our own loopback address, which is the caching-only server that we just started. Pointing to and using the local server is shown in Example 10-16.

Example 10-16 Local caching DNS specified, loopback address

```
BROWSE    TCPIPB.TCPPARMS(DATAB31N) - 01.02          Line 00000000 Col 001 080
TCPIPJOBNAME TCPIPB
HOSTNAME WTSC31
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  127.0.0.1   <=== Was 10.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

alloc ddn(systcpd) dsn('tcpipb.tcpparms(DATAB31N)') shr
nslookup

 EZB3170I Default Server:  localhost
 EZB3172I Address:  127.0.0.1
 EZB3042I >
wtsc30
 EZB3170I Server:  localhost
 EZB3172I Address:  127.0.0.1

 EZB3110I Non-authoritative answer:
 EZB3170I Name:    wtsc30.ITSO.IBM.COM
 EZB3172I Address:  10.12.4.211

 EZB3042I >
wtsc31
 EZB3170I Server:  localhost
 EZB3172I Address:  127.0.0.1

 EZB3110I Non-authoritative answer:
 EZB3170I Name:    wtsc31.ITSO.IBM.COM
 EZB3172I Address:  10.12.4.213

 EZB3042I >
```

Compare the server names and addresses between the two methods. Notice the first method used server *itsodns.itso.ibm.com* at *10.12.6.7*, but the second method used our *localhost* at

127.0.0.1. Also note that the caching-only server issued message `EZB3110I Non-authoritative answer`.

Next, use TSO DIG to query for the same names that were used for nslookup. Ours names were *wtsc30* and *wtsc31*, as shown in Example 10-17.

Example 10-17 DIG using loopback TCPDATA file, pointing to local loopback caching DNS

```
dig wtsc30

EZB3316I ;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 3
EZB3328I ;; Ques: 1,Ans: 1,Auth: 0,Addit: 3
EZB3332I ;; QUESTIONS:
;;      wtsc30.ITSO.IBM.COM, type = A, class = IN

EZB3359I ;; ANSWERS:
wtsc30.ITSO.IBM.COM.    A       10.1.1.10

dig wtsc31

EZB3316I ;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 3
EZB3328I ;; Ques: 1,Ans: 1,Auth: 0,Addit: 3
EZB3332I ;; QUESTIONS:
;;      wtsc31.ITSO.IBM.COM, type = A, class = IN

EZB3359I ;; ANSWERS:
wtsc31.ITSO.IBM.COM.    A       10.1.1.20
```

Browse the log to see the latest query activity just created

Checking the log again, as in Example 10-18, shows the results of the query activity (nslookup and dig) that was just performed.

Example 10-18 NAMED9 Log after using nslookup and dig for WTSC30 and WTSC31

```
BROWSE -- /SC31/etc/dnsdatb31/dns.log -------------- Line 00000000 Col 035 114

EZZ8828I client 127.0.0.1#1260: query: 1.0.0.127.in-addr.arpa IN PTR
EZZ8828I client 127.0.0.1#1261: query: wtsc30.ITSO.IBM.COM IN A
EZZ8828I client 127.0.0.1#1262: query: wtsc31.ITSO.IBM.COM IN A

EZZ8828I client 127.0.0.1#1289: query: . IN A
EZZ8828I client 127.0.0.1#1290: query: wtsc30.ITSO.IBM.COM IN A
EZZ8828I client 127.0.0.1#1296: query: wtsc31.ITSO.IBM.COM IN A
```

Stop the caching-only DNS server to verify it will shut down cleanly

When you stop the BIND 9 server using the MVS STOP (P) command, it logs shutdown messages, as shown in Example 10-19.

Example 10-19 NAMED9 log immediately after shutdown of NAMED9 task

```
BROWSE -- /SC31/etc/dnsdatb31/dns.log -------------- Line 00000000 Col 035 114

 1327ec20: EZZ9131I shutting down
 1327ec20: EZZ9045I no longer listening on 10.1.1.10#53
 1327ec20: EZZ9045I no longer listening on 10.1.1.20#53
 1327ec20: EZZ9045I no longer listening on 127.0.0.1#53
```

10.4 Automated domain name registration

The automated domain name registration (ADNR) application enables you to dynamically add or delete application-specific hostnames (and the addresses of those applications) in name servers according to application availability. The DNS names managed by ADNR can be names that represent a specific instance of an application within the sysplex, names that represent individual systems within the sysplex, and names that represent the entire sysplex.

This section includes the following topics:

- Description of ADNR
- Configuration of ADNR
- Activation and verification of ADNR

10.4.1 Description of ADNR

The name server or name servers that ADNR updates can be z/OS BIND 9 name servers, or non-z/OS name servers that support BIND 9 dynamic update. ADNR supports both IPv4 and IPv6 addresses. Figure 10-3 depicts the communication relationship between ADNR, the Load Balancing Advisor (LBA), and the LB Agents.

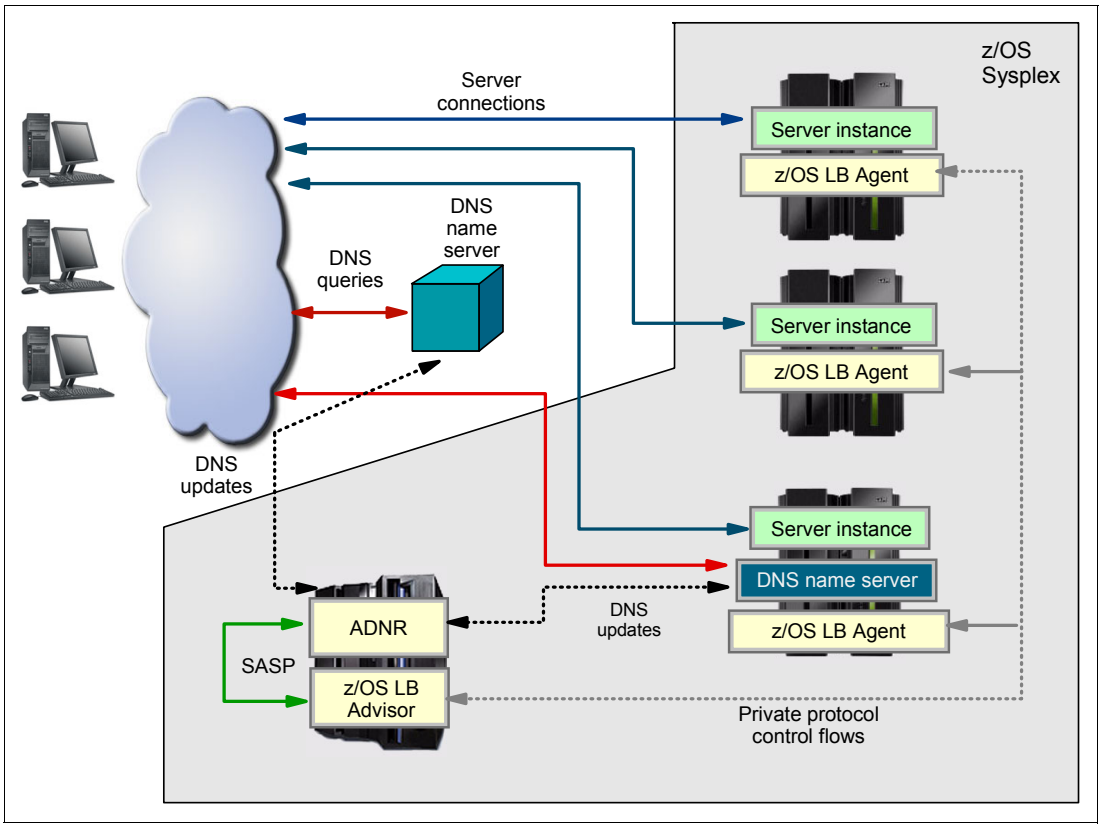


Figure 10-3 Diagram representing the flow among ADNR, Load Balancing Advisor, and agents

ADNR is configured with information about sysplex resources and their assigned DNS names. These resources are configured on a group basis. ADNR registers its configured information about sysplex resources with a z/OS Load Balancing Advisor application. The Advisor disseminates this information to the z/OS Load Balancing Agents, which report back to the Advisor about the availability of the resources registered by ADNR. The Advisor subsequently reports to ADNR any changes in the availability of those resources.

For each group of resources that the Advisor reports as available, ADNR adds a DNS name to the name server that represents the entire group of resources in that group, and maps that name to the IP addresses of the available resources in that group. The interaction between ADNR and the Load Balancing Advisor is through the SASP protocol (Server/Application State Protocol).

10.4.2 Configuration of ADNR

Note: You do not need to have Sysplex Distributor implemented in your environment to have ADNR configured.

For performance purposes, it is advantageous to run ADNR and LBA Advisor in the same image. The DNS server can be in the same image, too. Perform these tasks to configure ADNR:

- Create the ADNR configuration file
- Configure your TCPIP.DATA to point to the DNS server
- Configure the z/OS BIND9 DNS server file
- Create the LB Advisor configuration file
- Create the LB Agent configuration files for sysplex LPARs
- Define security server profiles for ADNR

Create the ADNR configuration file

Example 10-20 notes some basics elements of the ADNR configuration file.

Example 10-20 ADNR configuration file example

```
debug_level             128             # Error, Warning, Event, Info
uuid                    IBM_sysplex_adnr
dns                     dnsserv         # Label used by other stmts
                                        #  and commands
{
  dns_id                10.1.2.99..53   # Linux name server

# This zone will contain all addresses which may be used by intranet
# clients to reach applications within the sysplex.

  zone                  myzone
                                        # Label used by other stmts
                                        #  and commands
  {
    domain_suffix       example.com
                                        # Zone name in name server
#   update_key          mvsplex_update_key
                                        # Key to sign dyn. updates
#   transfer_key        mvsplex_transfer_key
                                        # Key for zone transfers
    ttl                 60              # Time To Live for zone
  } # end of zone
} # end of dns

#------------------------------
gwm                     z/os_lba_advisor
{
```

```
  gwm_id              10.1.9.30..3860 # LBA lb_connection_v4 address
  host_connection_addr  10.1.1.30       # Local address
} # end of gwm
# --------------------------ok--------------------------------
 host_group              itso_sysplex                              1

  {
  host_group_name       SC3Xgroup                                 2
  dns                   dnsserv
  zone                 example.com

  member                SC31                                      3
  {
    host_name           WTSC31        # Prepended to domain suffix  4
    ipaddrlist          sc31_vipa_addrs                           5
  }
  } # end of host_group definition
ipaddrlist              sc31_vipa_addrs                           5
  {
    ipaddr                10.1.1.20       # static VIPA on sc31     6
  } # end of ipaddrlist
```

In this example, the numbers correspond to the following information:

1. *Host_group* statement represents the mapping of host name-to-IP address in the DNS server.
2. *Host_group_name* is the name of the group of hosts to be updated in the name server.
3. The statement *member* defines a member for a given group.
4. You should define the name of the individual host to be updated in the name server through the *host_name* parameter.
5. The *ipaddrlist* parameter is associated with the member host_name.
6. The *ipaddr* parameter is the static VIPA address representing the stack.

In our configuration we used a basic customization, including just one IP address to monitor, as shown in Example 10-20 on page 380. When you define a member in the ADNR configuration, you are saying that this member should be monitored by the Agents and if this address does not respond the Agent will inform the Advisor. The Advisor then informs ADNR so it can proceed with an update into the DNS server. Our configuration is not using authentication, so the statements *key*, *transfer-key*, and *update_key* are not used. Use of these keys causes the data sent by ADNR to be signed by ADNR and authenticated by the name server. If you need the authentication function, you can get more informations about how to configure it in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Configure your TCPIP.DATA to point to the DNS server

Example 10-21 shows the TCPIP.DATA file.

Example 10-21 TCPIP.DATA file configuration

```
TCPIPJOBNAME TCPIPB
HOSTNAME WTSC30B
DOMAINORIGIN  EXAMPLE.COM  1
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  10.1.2.99    2
```

```
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
```

In this example, the numbers correspond to the following information:

1. The domain should match the zone defined on the DNS server.
2. The Name Server IP address must be the IP address of the DNS server.

Configure the z/OS BIND9 DNS server file

Example 10-22 shows our DNS server configuration file (`named.conf`).

Example 10-22 DNS server config file for example.com

```
[root@dnsserv:etc] # more named.conf
// named.conf for example.com
logging {
     channel dnslog {
          file "/var/named/bind9.log" versions 3 size 3m;
          severity debug 10;
          print-time yes;
          print-severity yes;
          print-category yes;
          };
     category default {
          dnslog;
          };
     };
options {
     directory "/var/named";
     dump-file "/var/named/data/cache_dump.db";
     statistics-file "/var/named/data/named_stats.txt";
     version "9.2.4";
};
zone "example.com" in {
     type master;                                          1
     file "example.com.zone";                              3
     allow-update { any; };                                2
     };
```

In this example, the numbers correspond to the following information:

1. The name servers that ADNR manages require a one-time setup. These name servers must be configured as the primary master name servers for the zones managed by ADNR.
2. We configured our zone statements with “allow-update { any; };”, because the DNS server was in our test environment. This however, is an unsafe practice in a production environment. It allows the zone to be updated by unauthorized personnel or programs and could present a security risk.
3. Zones managed by ADNR must be updated exclusively by ADNR. The zones and zone files should not be edited by hand nor should anyone or any other program perform dynamic updates to the zones.

Create the LB Advisor configuration file

You must configure the Load Balancing Advisor, as shown in Example 10-23. An external load balancer is not required.

Example 10-23 Load Balancing Advisor configuration file

```
#Load Balancing Advisor Configuration
debug_level               7
update_interval           60
agent_connection_port     8100

agent_id_list
{
  10.1.1.20..8000
  10.1.1.10..8000
}
lb_connection_v4          10.1.9.30..3860
lb_id_list
{
  10.1.1.30
}
wlm serverwlm
```

Create the LB Agent configuration files for sysplex LPARs

Example 10-24 shows the LB Agent configuration files for SC30.

Example 10-24 Load Balancing Agent configuration file for SC30

```
debug_level               7
advisor_id                10.1.1.30..8100
host_connection           10.1.1.10..8000
```

Example 10-25 shows the LB Agent configuration file for SC31.

Example 10-25 Load Balancing Agent configuration file for SC31

```
debug_level               7
advisor_id                10.1.1.30..8100
host_connection           10.1.1.20..8000
```

Define security server profiles for ADNR

Copy the ADNR start procedure from SEZAINST(EZBADNRS) and customize it, as shown in Example 10-26.

Example 10-26 Start procedure example for ADNR

```
BROWSE    SYS1.PROCLIB(ADNR) - 01.05
//ADNR     PROC
//ADNR   EXEC PGM=EZBADNR,REGION=OK,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)/',
//            'ENVAR("_CEE_ENVFILE=DD:STDENV")/')
//CONFIG   DD DSN=TCPIPC.TCPPARMS(ADNRCNFC),DISP=SHR
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(ADNR&SYSCLONE.)
//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=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//CEESNAP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSMDUMP DD DUMMY
```

Define RACF USERID, STARTED, OPERCMD profiles for ADNR start procedure, and grant authority to start ADNR as follows:

```
ADDUSER ADNR DFLTGRP(OMVSGRP) OMVS(UID(9999) HOME('/tmp') PROGRAM('/bin/sh'))
RDEFINE STARTED ADNR.* STDATA(USER(ADNR))
SETROPTS CLASSACT(OPERCMDS)
SETROPTS RACLIST (OPERCMDS)
RDEFINE OPERCMDS (MVS.SERVMGR.ADNR) UACC(NONE)
PERMIT MVS.SERVMGR.ADNR CLASS(OPERCMDS) ACCESS(CONTROL) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

ADNR must have read and write access to the directory specified on the HOME keyword. This directory becomes ADNR's *working directory*. ADNR creates and deletes temporary files in this directory during its operation. The UID value, nn, can be zero or nonzero. Also, the program specified on the user ID assigned to run ADNR must be `/bin/sh`.

Note: ADNR is a multithreaded application. If you define an unusually large number of name servers or zones to ADNR, you should check to determine whether the maximum number of threads allowed per process, represented by the THREADSMAX value in BPXPRMxx, is going to be exceeded. The number of threads required for ADNR is determined in the following way: (number of dns statements) + (number of zone keywords within all dns statements) + 3. You can customize the maximum number of threads allowed for ADNR by specifying the THREADSMAX keyword on the ADDUSER command.

10.4.3 Activation and verification of ADNR

After ADNR registers its resources with the Advisor, it waits for a period of time to ensure it has received all information from the Advisor about the availability of these resources. The *convergence interval* is normally two times the Advisor update interval. ADNR issues an event message when the convergence is complete.

To determine whether convergence is complete, display ADNR's GWM state as shown in Example 10-27, using the command **MODIFY procname,DISP,GWM**.

Example 10-27 Example showing the convergence between ADNR and Load Balancing Advisor

```
F ADNR,DISPLAY,GWM
EZD1254I GWM SUMMARY 663
GWM LABEL        : Z/OS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
1 OF 1 RECORDS DISPLAYED
```

After the convergence completes, ADNR attempts to update all the name server's zones with the appropriate resource records. Records for resources that are no longer available are update-deleted from each zone and those that are available are update-added to each zone if they are not already present.

After this has occurred, the zone is in SYNCHRONIZED state and you can check it using the command **MODIFY procname,DISP,DNS,ZONES**.

ADNR examines the contents of zones specified on the `dns` statement during zone resynchronization.

We used the following steps to verify that when the TCP/IP in LPAR SC31 becomes unavailable its name (sc31.example.com) is removed from the DNS server. After restarting the TCPIP in that LPAR the name should be registered again on the DNS server. The steps we used are:

- Verify you can ping the sysplex member DNS name.
- Stop the TCP/IP stack in the sysplex member.
- Verify that the DNS status has changed for ADNR.
- Verify that you cannot ping the sysplex member DNS name.
- Restart the TCP/IP stack and display the zones again.
- Verify you can ping the sysplex member DNS name again.

Verify you can ping the sysplex member DNS name

We issued the following command from SC32:

```
ping WTSC31.example.com
```

We received the output shown in Example 10-28.

Example 10-28 Ping from the SC32 system against the name registered on DNS server through ADNR

```
Pinging host WTSC31.EXAMPLE.COM (10.1.1.20)
Ping #1 response took 0.000 seconds.
```

Stop the TCP/IP stack in the sysplex member

Stop the TCP/IP stack on the sysplex member (in our case: SC31). This action will cause its IP address to become unavailable.

Verify that the DNS status has changed for ADNR

Issue the following console command:

```
F ADNR,DISP,DNS,ZONES,DETAIL
```

The display shows that the label of the sysplex member (in our case: SC31) is removing from the DNS server (see Example 10-29).

Example 10-29 Display of the zones on DNS when removing the TCP/IP stack

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 915
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/01/10 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : UPDATE-DELETE_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
```

```
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/02/10 19:05:47
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : UPDATE-DELETE_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/02/10 19:05:47
```

The display shows that the label of the sysplex member (in our case: SC31) is not present anymore (see Example 10-30).

Example 10-30 Display of the zones on DNS

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 858
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/02/10 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : NOT_PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/02/10 18:58:26
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : NOT_PRESENT          1
   TTL           : 60
   CLASS         : IN
   TYPE          : A
  RDATA         : 10.1.1.20
  GWM LABEL     : ZOS_LBA_ADVISOR
  GROUP LABEL   : ITSO_SYSPLEX
  LAST UPDATE   : 10/02/10 18:58:26
```

In this example, the current status, NOT_PRESENT, indicates that there is no connectivity to SC31, as shown at 1.

Verify that you cannot ping the sysplex member DNS name

Try a **ping** command again (from SC32 in our environment):

```
ping WTSC31.example.com
```

Example 10-31 shows the output of this command.

Example 10-31 Ping with no success after we stopped the TCPIPC on SC31

```
ping WTSC31.example.com
EZZ3111I Unknown host 'WTSC31.EXAMPLE.COM'
```

Restart the TCP/IP stack and display the zones again

Start the TCP/IP stack on the sysplex member (in our case: SC31). This action causes the IP address to become available. After starting the TCP/IP stack, issue the following command:

```
F ADNR,DISP,DNS,ZONES,DETAIL
```

You might initially see output similar that shown in Example 10-32. The process takes several seconds to update the DNS server and you might see `Update-Add_in_Progress` as intermediate status.

Example 10-32 Starting the TCP/IP stack and issuing display of the DNS RR status

```
S TCPIPB
$HASP100 TCPIPB ON STCINRDR
IEF695I START TCPIPB WITH JOBNAME TCPIPB IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TCPIPB STARTED
EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPB ARE AVAILABLE.

F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 909
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/02/10 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : UPDATE-ADD_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/02/10 18:58:26
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : UPDATE-ADD_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
```

```
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/02/10 18:58:26
```

If you issue the command again after a short time you should see the results similar to that shown in Example 10-33.

Example 10-33 DNS RR status changes to PRESENT

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 847
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/02/10 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : N/A
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : N/A
```

Verify you can ping the sysplex member DNS name again

We issued the following command from SC32:

```
ping WTSC31.example.com
```

Example 10-34 shows the output of this command.

Example 10-34 Ping from the SC32 system against the name registered on DNS server through ADNR

```
Pinging host WTSC31.EXAMPLE.COM (10.1.1.20)
Ping #1 response took 0.000 seconds.
```

10.5 Problem determination for DNS service

This section covers the problem determination procedures for the caching-only DNS server and those for ADNR.

10.5.1 Problem determination for a caching-only DNS server

DNS server problems can be managed by using the following procedures:

- Review startup error messages.
- Review syslogd to obtain early logging messages.
- Start named from the OMVS shell to obtain early logging messages.
- Set debug and trace levels.
- Use the OMVS rndc command to manage the DNS daemon.

Review startup error messages

BPXBATCH does not handle a split parameter string that is created by putting single quotes around multiple JCL records and continuing them with a comma. If the PARM string is long enough to be continued onto the next line, you must code the parameter string contiguously to column 71, placing a continuation character in column 72, and then continuing the parameter string in column 16 of the next line. The unsupported format and resulting error message is shown in Example 10-35.

Example 10-35 NAMED9 JCL: example of invalid PARM string and resulting message at startup

```
BROWSE    SYS1.PROCLIB(NAMED9B) - 01.00             Line 00000000 Col 001 080
//NAMED9B  PROC PARMS='-d 0'
//NAMED9B  EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
// PARM=('PGM /usr/lpp/tcpip/sbin/named -V v9 ',
//       ' -c /etc/dnsdatb&SYSCLONE./named9b.conf &PARMS')
//STDENV   DD PATH='/etc/dnsdatb&SYSCLONE./named9b.env',
//            PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*

J E S 2  J O B  L O G  --  S Y S T E M  S C 3 1 --  N O D E
20.34.19 STC08336 ---- MONDAY,    24 Sep 2007 ----
20.34.19 STC08336  IEF695I START NAMED9X  WITH JOBNAME NAMED9X  IS ASSIGNED TO U
20.34.19 STC08336  $HASP373 NAMED9X  STARTED
20.34.20 STC08336  EZZ9089I EXITING NAMED, BIND V9 (DUE TO EARLY FATAL ERROR)
20.34.20 STC08336  $HASP395 NAMED9X  ENDED

The correct PARM field should look like this:
// PARM=('PGM /usr/lpp/tcpip/sbin/named -V v9 -c /etc/dnsdatb&SYSCLONE.X
//             /named9b.conf &PARMS')
```

Review syslogd to obtain early logging messages

For the BIND 9 name server, initial startup messages go to syslog. Later messages will be directed to other defined or default logs according to logging statements found or implied in

the configuration file. For descriptions of the syslog file and the syslog daemon, see Chapter 1, “The syslog daemon” on page 1.

Start named from the OMVS shell to obtain early logging messages

The USERID that starts NAMED from the shell must have UID(0). Make sure your user ID either has UID(0) or that you can switch to superuser by entering the **su** command before you start NAMED.

If in your environment, your NAMED task requires environment variables, you can use the same environment variables file in the shell that is used by the started task JCL. Simply use the **export** command to set the file name containing the environment variables.

When you start **named** from the shell, you can specify a parameter of **-g** to ask for detailed logging. This can be beneficial if BPXBATCH or NAMED have problems initializing. You might miscode a parameter, or use invalid syntax on one or more of the configuration statements, which causes either of the programs to terminate before they have progressed far enough to understand your logging options and directives. Use of the **-g** parameter overrides any logging (default or explicit) directives, and forces messages to be displayed to your shell session.

The use of the **su**, **export**, and the **named** commands is shown in Example 10-36.

Example 10-36 Starting NAMED9 from the z/OS UNIX shell to get early logging information

```
su

export _CEE_ENVFILE="/etc/dnsdatb31/named9b.env"

named -V v9 -g -c /etc/dnsdatb31/named9b.conf
```

Set debug and trace levels

Debugging levels can be specified at NAMED startup by using the **-d** parameter. They can later be changed in the configuration file, causing the server to reread the file. The BIND 9 name server relies on a start option, **rndc**, or the configuration file to define and alter the debug level. You can change the logging options in named.conf to gather more information, and then issue the SIGHUP signal or **rndc reload** to have the new logging options take effect. However, the preferred method is by using **rndc trace level**.

Use the OMVS rndc command to manage the DNS daemon

The **rndc** utility can be used to provide a variety of functions that can be helpful in debugging name server problems. For example, the name server’s cache can be viewed using the *dumpdb* parameter, and debug trace can be turned on or off using the *trace* parameter. If you suspect your cache is corrupted, you can flush the name server’s cache with the *flush* parameter.

For complete details about using the **rndc** command, see *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781.

10.5.2 Problem determination for ADNR

If there are problems with ADNR, try the following problem determination techniques:

- Verify that DNS, LBA and ADNR are running
- Verify that the GWM is active
- Check that the sysplex members are available
- Verify that the DNS status is active
- More problem determination tests for ADNR

Verify that DNS, LBA and ADNR are running

Verify that the started task of Load Balance Advisor and ADNR are running. In addition, verify that the DNS server is running.

Verify that the GWM is active

Issue the following console command:

```
F ADNR,DISP,GWM,DETAIL
```

Example 10-37 shows the output of this command.

Example 10-37 Display the GWM status

```
F ADNR,DISP,GWM,DETAIL
...
 EZD1254I GWM DETAIL 953
 GWM LABEL        : ZOS_LBA_ADVISOR
  GWM STATUS      : GWM_ACTIVE
  GWM TIMESTAMP   : 10/02/10 18:51:39
  GWM IPADDR..PORT: 10.1.9.29..3860
  LOCAL IPADDR    : 10.1.1.30
  UUID            : IBM_SYSPLEX_ADNR
  UPDATE INTERVAL : 60
  LAST UPDATE     : 10/02/10 19:07:47
 1 OF 1 RECORDS DISPLAYED
```

You might initially see the GWM STATUS value as CONVERGENCE_PENDING, but after a short time the status will change to GWM_ACTIVE, if ADNR is functioning. When the status changes to GWM_ACTIVE, it means that any updates from the GWM are reflected in the name server.

Check that the sysplex members are available

Issue the following console command:

```
F ADNR,DISP,GROUPS,GROUPID=itso_sysplex,DETAIL
```

This should produce results similar to those shown in Example 10-38.

Example 10-38 Display of the GROUP members availability status

```
F ADNR,DISP,GWM,GROUPS,GROUPID=ITSO_SYSPLEX,DETAIL
EZD1254I GWM GROUP DETAIL 950
GWM LABEL        : ZOS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
 GWM TIMESTAMP   : 10/02/10 18:51:39
 GWM IPADDR..PORT: 10.1.9.29..3860
 LOCAL IPADDR    : 10.1.1.30
```

```
 UUID            : IBM_SYSPLEX_ADNR
 UPDATE INTERVAL : 60
 LAST UPDATE     : 10/02/10 19:07:47
 GROUP LABEL     : ITSO_SYSPLEX
  GROUP NAME     : SC3XGROUP
  GROUP TYPE     : HOST
  DNS LABEL      : TEST85DNS
  ZONE LABEL     : MYZONE
  MEMBER HOSTNAME:
   IPADDR        : 10.1.1.20
    AVAIL        : NO
    FLAGS        : NOTARGETSYS
    UPDATE COUNT : 4
  MEMBER HOSTNAME: WTSC31
   IPADDR        : 10.1.1.20
    AVAIL        : NO
    FLAGS        : NOTARGETSYS
    UPDATE COUNT : 4
```

The name is registered with the DNS server is WTSC31.example.com, where the first part of the name (WTSC31) came from the member host name. The remainder of the name is from domain_suffix defined on the zone statement as shown in Example 10-20 on page 380.

The *itso_sysplex* is the host_group name from the ADNR configuration file. Verify that under MEMBER HOSTNAME, the AVAIL field has the value YES. If the value is NO, check the configuration file of ADNR and update the IPADDR statement of the IP address which is not available to have an IP address that is configured under in the LBA configuration file.

Note: To enable ADNR to connect to the Advisor, the source IP address that ADNR uses to connect to the Advisor must be configured in the Advisor's lb_id_list statement. For information about the lb_id_list statement, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Verify that the DNS status is active

Issue the following console command:

```
F ADNR,DISP,DNS,DETAIL
```

The results are similar to Example 10-39. This provides information about the name server that ADNR is updating. The DNS status should be ACTIVE, and you should have at least one active zone.

Example 10-39 Displaying the DNS status

```
F ADNR,DISP,DNS,DETAIL
EZD1254I DNS DETAIL 915
DNS LABEL        : DNSSERV
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
1 OF 1 RECORDS DISPLAYED
```

More problem determination tests for ADNR

If the ADNR process does not work, try the following steps to find the problem:

1. Enable high level of logging for ADNR by issuing the following command:

   ```
   F ADNR,DEBUG,LEVEL=32
   ```

2. Verify that you get messages sent to syslogd:

   a. Verify that syslogd is up by issuing **ps –ef | grep syslogd** under z/OS UNIX.

   b. Verify that the configuration file for syslogd is configured correctly.

   c. If you make any change to the syslogd configuration file, stop and start syslogd. You can do it by issuing a **kill** command stop syslogd and issuing the syslogd command to start syslogd. By default it uses the /etc/syslog.conf configuration file.

   d. Verify that you have enough free space in the directory where the log file resides.

   Note: The ADNR log file is helpful for solving problems.

3. Verify that the DNS server version is at least 9.2. You can do it by issuing the following command from the DNS server:

   ```
   named –v
   ```

4. Enable logging on your DNS server by updating the DNS configuration file (usually it will be /etc/named.conf) to include the line shown in Example 10-40.

   Example 10-40 Enabling high logging level on the DNS server

   ```
   logging {
        channel dnslog {
             file "/var/named/bind9.log" versions 3 size 3m;
             severity debug 10;
             print-time yes;
             print-severity yes;
             print-category yes;
             };
        category default {
             dnslog;
             };
        };
   ```

   In this example the log of the DNS is written to /var/named/bind9.log. After you update the configuration file, restart the DNS server.

5. From the DNS server, issue the following command:

   ```
   tail -f /var/named/bind.log
   ```

   This command displays the last lines of the log file and outputs appended data as the file grows. You can use this command to see if the z/OS image is trying to send requests to the DNS server. You should leave the tail command running till you are satisfied that ADNR is working.

6. Verify that you can issue **nslookup** from the z/OS image to the specific DNS server by issuing the following command:

   ```
   nslookup
   ```

   From the **nslookup** program issue the following command:

   ```
   server 10.1.2.99
   ```

In this command, `10.1.2.99` represents the DNS server IP address.

The following information displays:

```
   Default Server:  dnsserv.example.com
   Address:  10.1.2.99
>
```

Type `exit` to end the **`nslookup`** program.

Verify that the user who runs the ADNR started task has full permissions on its home directory. If the permissions are insufficient you might see the message shown in Example 10-41.

Example 10-41 Console error message

```
BPXF903I THE ATTRIBUTE RETRIEVAL CALL (IGWASMS) FOR FILE SYSTEM 576
.DIGRC.HFS FAILED.
RC = 00000008, RS = 00000008, DIAG = 0000000000000000
```

In addition, you will see the message shown in Example 10-42 in the ADNR log file.

Example 10-42 Error message in the ADNR log file

```
Zone transfer for zone myzone failed, rc = No file could be created
```

10.6 Additional information sources for DNS

See the following sources for additional information about the DNS server and protocol:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998

Tip: For descriptions of security considerations that affect specific servers or components, see "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

A

Environment variables

This appendix discusses environment variables used by the z/OS TCP/IP stack and associated applications.

Environment variables are named variables, with assigned values, that can be accessed by various processes in the z/OS Communications Server configuration. Applications use environment variables to define the characteristics of their specific environment. *z/OS Communications Server: IP Configuration Guide*, SC31-8775, provides details about where to find information about the environment variables that are explicitly set by the z/OS Communications Server and its applications. In addition to an application being able to set its own environment variables, Language Environment and z/OS UNIX System Services (z/OS UNIX) also provide environment variables. The following discussion assists with determining which applications have access to these additional Language Environment and z/OS UNIX System Services variables.

Understanding the resolver search orders used in native MVS API environments and in z/OS UNIX environments is key to setting up your system properly. The type of API environment not only affects what search order is used by the resolver to locate certain files required for processing, but it also determines which set of environment variables is available to the resolver and to the server programs. You can indirectly determine which sets of environment variables an application can use by identifying the *caller API value* obtained from the output of a resolver trace performed when the application calls the resolver. For information about dynamically starting the resolver trace, see *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

Description of the environment variable information

We summarized many details by organizing the information into tables. Table A-1 and Table A-2 on page 397 show sample commands and applications that use the various API interfaces. They list the most commonly used environment variables for the z/OS TCP/IP stack and associated applications. Some applications use only the z/OS UNIX System Services API, some use only the Language Environment API, some use both, and some use neither. In addition, some applications set their own environment variables.

The variables associated with the z/OS UNIX System Services API are listed in Table A-3 on page 397. The variables associated with the Language Environment API are listed in Table A-4 on page 398.

The variables specific to each application are listed in Table A-5 on page 399. The application-specific table also indicates which of the API interfaces are used by the application. When an application uses the indicated API interface, it also has access to that API's set of environmental variables.

The following sections discuss the API environments:

- Native MVS API environment
- z/OS UNIX API environment
- z/OS UNIX System Services environment variables
- Language Environment variables
- Application-specific environment variables
- Setting environment variables

Native MVS API environment

The following *caller API values* indicate that the native *MVS API environment* search order is used, and access to the Language Environment and z/OS UNIX System Services environment variables is not available:

- TCP/IP C Sockets
- TCP/IP Pascal Sockets
- TCP/IP REXX Sockets
- TCP/IP Sockets Extended

Table A-1 lists examples of commands and applications that use the native *MVS API environment*.

Table A-1 Examples of commands and applications that use the native MVS API environment

TSO commands	Applications
DIG LPR NETSTAT NSLOOKUP PING REXEC RPCINFO RSH TRACERTE	CICS Listener LPD Miscellaneous server PORTMAP RSHD (RXSERVE) SMTP TN3270

z/OS UNIX API environment

The following *caller API values* indicate that the z/OS UNIX API environment search order is used, and the Language Environment and z/OS UNIX System Services environment variables are available:

- Language Environment C Sockets
- z/OS UNIX

Table A-2 lists some examples of commands and applications that use the z/OS UNIX environment.

Table A-2 Examples of commands and applications that use the z/OS UNIX environment

z/OS UNIX command	Applications
dig dnsdomainname domainname ftp host hostname netstat nslookup ping rexec rsh rpcinfo sendmail snmp traceroute	FTP SNMP agent z/OS UNIX OPORTMAP z/OS UNIX OREXECD z/OS UNIX ORSHD z/OS UNIX RPCBIND

z/OS UNIX System Services environment variables

z/OS UNIX System Services Planning, GA22-7800, contains an appendix that discusses environment variables (_BPX_ and _BPXK_) used by the z/OS UNIX System Services kernel. Applications that use the z/OS UNIX System Services API have access to these variables. These are not specific to TCP/IP functions and are listed here for completeness.

Table A-3 z/OS UNIX System Services API environment variables

z/OS UNIX System Services API environment variable	Description
_BPX_ACCT_DATA	Sets accounting information for the caller
_BPX_BATCH_SPAWN	Specifies whether BPXBATCH is to use spawn instead of fork
_BPX_BATCH_UMASK	Sets the permission bits for a file
_BPX_JOBNAME	Sets the job name of a process by overriding the default
_BPX_PTRACE_ATTACH	Used for debugging target programs
_BPX_SHAREAS	Allows a spawned child process to share the shell's address space
_BPX_SPAWN_SCRIPT	Flags the specified file as a shell script
_BPX_TERMPATH	Determines the origin of a logged user

_BPX_UNLIMITED_SPOOL	Can limit spooled output
_BPX_USERID	Sets the user ID of a spawned process
_BPXK_AUTOCVT	Controls automatic file conversion
_BPXK_CCIDS	Defines a pair of coded character set IDs
_BPXK_DAEMON_ATTACH	Controls the security environment of RACF-DELEGATED resources
_BPXK_INET_FASTPATH	Used by TCP/IP to access FASTPATH mode
_BPXK_JOBLOG	Controls writing WTO messages to a job log file
_BPXK_MDUMP	Controls destination of SYSMDUMP output
_BPXK_SETIBMOPT_TRANSPORT	Sets stack affinity
_BPXK_WLM_PROPAGATE	Controls propagation of WLM enclaves

Language Environment variables

z/OS XL C/C++ Programming Guide, SC09-4765, contains information about some of the environment variables reserved for z/OS XL C/C++. Applications that use the Language Environment API have access to these variables.

Table A-4 Language Environment API environment variables

Language Environment variable	Description
_CEE_DLLLOAD_XPCOMPAT	Controls z/OS DLL load order
_CEE_DMPTARG	Specifies directory for Language Environment dumps (CEEDUMPs)
_CEE_ENVFILE	Points to a file containing other environment variables
_CEE_HEAP_MANAGER	Specifies the Vendor Heap Manager (VHM) DLL name
_CEE_RUNOPTS	Sets Language Environment runtime options
_EDC_ADD_ERRNO2	Appends errno2 information to the output of perror() and strerror()
_EDC_ANSI_OPEN_DEFAULT	Affects characteristics of z/OS text files opened with default attributes
_EDC_BYTE_SEEK	Indicates that ftell() should return relative byte offsets
_EDC_CLEAR_SCREEN	Affects the clearing of output screens
_EDC_COMPAT	Controls compatibility with old IBM C/370™ code
_EDC_C99_NAN	Controls binary floating-point representation
_EDC_ERRNO_DIAG	Controls additional diagnostic information
_EDC_GLOBAL_STREAMS	Controls the stdin, stdout, and stderr data streams
_EDC_POPEN	Uses fork() or spawn() to create a child process
_EDC_PUTENV_COPY	Sets the behavior of the putenv() function
_EDC_RRDS_HIDE_KEY	Applies to VSAM RRDS files opened in record mode
_EDC_STOR_INCREMENT	Controls storage increments above the 16M line
_EDC_STOR_INCREMENT_B	Controls storage increments below the 16M line

_EDC_STOR_INITIAL	Sets initial size of library storage above the 16M line
_EDC_STOR_INITIAL_B	Sets initial size of library storage below the 16M line
_EDC_ZERO_RECLEN	Allows processing of zero-length records in a z/OS variable file

Application-specific environment variables

Table A-5 lists the application-specific environment variables.

The first table entry for each application indicates whether the application has access to the z/OS UNIX API variables referenced in Table A-3 on page 397. When the z/OS UNIX API variables entry indicates `Yes`, the application has access to the z/OS UNIX API environment variables.

The second table entry for each application indicates whether the application has access to the Language Environment API variables referenced in Table A-4 on page 398. When the Language Environment API variables entry indicates `Yes`, the application has access to the Language Environment API environment variables.

The remaining table entries for each application show the application's specific environment variables.

Table A-5 Application-specific environment variables

Application-specific environment variable	Description
RESOLVER	
z/OS UNIX API variables	Has indirect access to these using program call
Language Environment API variables	Has no access to these
RESOLVER_IPNODES	Points to the /etc/ipnodes file
X_ADDR	Points to hlq.HOSTS.ADDRINFO
X_SITE	Points to hlq.HOSTS.SITEINFO
HOSTALIASES	Points to the host alias name file
RESOLVER_TRACE	Points to the file into which the resolver trace output is written
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
X_XLATE	Points to hlq.STANDARD.TCPXLBIN
General stack environment configuration	
z/OS UNIX API variables	Has indirect access using program call
Language Environment API variables	Has indirect access using program call
LOCALDOMAIN	Overrides any other setting for domain within TCPDATA
MESSAGECASE	Overrides any other setting for message case within TCPDATA

SYSLOGD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
IKED	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
IKED_FILE	Configuration file for IKED
IKED_CTRACE_MEMBER	Parmlib member containing CTRACE settings for IKED
IKED_CODEPAGE	EBCDIC single-byte code page
Load Balancing Advisor and Agent	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
TN3270E Telnet server	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 397
Language Environment API variables	No access to the variables in Table A-4 on page 398
INETD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
OTELNETD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
KRB5_SERVER_KEYTAB	Kerberos security setting
LC_ALL	Sets all the above LC_ variables with one setting
LC_COLLATE	Character collation
LC_CTYPE	Character handling
LC_MESSAGES	Message handling
LC_NUMERIC	Numeric formatting
LC_TIME	Date and time formatting
NLSPATH	Locates message catalogs
TERMINFO	Unsupported terminal
FTP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
KRB5_SERVER_KEYTAB	Kerberos security setting

_FTPXLATE_name	(CCXLATE name) Translate table for control connection
_FTPXLATE_name	(XLATE name) Translate table for data connection
_ICONV_UCS2	Conversion methods for FTP
TFTPD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
TSO REXEC	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 397
Language Environment API variables	No access to the variables in Table A-4 on page 398
UNIX REXECD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
UNIX RSHD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
_EUV_SVC_STDOUT_FILENAME	Sets default message output file name
_EUV_SVC_DBG_FILENAME	Overrides default file: Debug messages written to this file
_EUV_SVC_DBG_MSG_LOGGIN	Specifies whether to generate debug message
_EUV_SVC_DBG_TRACE	Specifies whether to generate trace messages
_EUV_SVC_MSG_FACILITY	Facility class setting
_EUV_SVC_MSG_LOGGING	Logging method
KRB5_SERVER_KETAB	Kerberos security table
LANG	Local language and customs
LC_COLLATE	Character collation
LC_CTYPE	Character handling
LC_MESSAGES	Message handling
LC_MONETARY	Currency formatting
LC_NUMERIC	Numeric formatting
LC_TIME	Date and time formatting
LC_ALL	Sets all the above LC_ variables with one setting
NLSPATH	Locates message catalogs
PATH	Default path(s) for locating files

SMTP	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 397
Language Environment API variables	No access to the variables in Table A-4 on page 398
CSSMTP	
CSSMTP_CODEPAGE_CONFIG	EBCDIC single-byte code page
Sendmail	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
GSK_TRACE	Specifies a bit mask enabling system SSL trace options.
GSK_TRACE_FILE	When set to the name of a file, enables the system SSL trace.
HOME	The path name of the user's home directory.
HOSTALIASES	The host aliases file for sendmail.
LPD/LPR	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 397
Language Environment API variables	No access to the variables in Table A-4 on page 398
DHCP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
LPR_PRINTER	Output printer for DHCP reports
BIND4	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
DNS_VERSION	Version of Bind (V4 or V9) Default setting
HOSTALIASES	Host aliases file
PAGER	VIEW subcommand filter
BIND9	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
DNS_VERSION	Version of Bind (V4 or V9) Default setting
OMPROUTE	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
OMPROUTE_FILE	Configuration file for Omproute
OMPROUTE_DEBUG_FILE	Debug output file for Omproute

OMPROUTE_DEBUG_FILE_CONTROL=1000,5	Omproute's Debug file size
OMPROUTE_OPTIONS=hello_hi	Options for Omproute
OMPROUTE_CTRACE_MEMBER=CTIORAT0	CTRACE options for Omproute
SNMP_PORT	SNMP subagent listens on this port
TMPDIR	Directory for temporary work files
SNMP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
OSNMP_CONF	Configuration file for **osnmp**/**snmp** command /etc/snmpv2.conf
SNMPD_CONF	Configuration file for SNMP agent (V3) /etc/snmpd.conf
MIBS_DATA	MIB descriptions for SNMP /etc/mibs.data
OSNMPD_DATA	Control file for SNMP agent /etc/osnmpd.data
PW_SRC	Community security file for SNMP (V2) /etc/pw.src
SNMPD_BOOTS	Boot file for SNMPD agent (v3) /etc/snmpd.boots
SNMPTRAP_DEST	Control file for SNMPTRAP (v2) /etc/snmptrap.dest
TRAPFWD_CONF	Configuration file for TRAPFWD /etc/trapfwd.conf
SNMPQE	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
MIB_DESC	MIB description file for SNMPQE /etc/mibdesc.data
SLAP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
SNMP_PORT	SNMP agent listens on this port
Policy Agent	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
LIBPATH	Program execution library /usr/lib
TZ	Local time zone
PAGENT_CONFIG_FILE	Configuration file for policy agent /etc/pagent.conf
PAGENT_LOG_FILE	Logfile for policy agent
PAGENT_LOG_FILE_CONTROL	Policy Agent's logfile size

RSVP Agent	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
RSVPD_CONFIG_FILE	Configuration file for RSVP /etc/rsvpd.conf
TRMD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
TZ	Local time zone
LIBPATH	Program execution library /usr/lib
TIMED	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
SNTPD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
PORTMAPPER	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
UNIX PORTMAPPER	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
MISCSERV	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
DCAS	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 397
Language Environment API variables	Has access to the variables in Table A-4 on page 398
DCAS_CONFIG_FILE	Configuration file /etc/dcas.conf for DCAS

Setting environment variables

Setting an environment variable so that a z/OS UNIX application can 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 /etc/tcpa.data, you can code the following export command:

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

If, instead of a UNIX file, you want to set RESOLVER_CONFIG to the data set MVSA.PROD.PARMS(TCPDATA), you can specify the following export command. Be certain to put the single quotation marks (' ') around the data set name. If you do not, your user ID will be added as a prefix to the data set name when the resolver 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 must be passed as a parameter in the JCL for the application. For example, the following lines show the RESOLVER_CONFIG variable pointing to a UNIX file:

```
//OSNMPD PROC
//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 pointing to a PDS member:

```
//OSNMPD    PROC
//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 the RESOLVER_CONFIG variable pointing to the TCPIP.DATA information from a DD card:

```
//OSNMPD PROC
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/',
// 'ENVAR("RESOLVER_CONFIG=DD:TCPDATA")/-d 0')
//TCPDATA DD DSN=TCPA.MYFILE(TCPDATA),DISP=SHR
```

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

```
//OSNMPD    PROC
//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(ENVFILE),DISP=SHR
```

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

Important: If this file is a z/OS data set (traditionally referred to as an MVS data set), the data set must be allocated with RECFM=V.

Using RECFM=F is *not* recommended, because RECFM=F enables padding with blanks for the environment variables. In this case, if an environment variable is specifying a UNIX file name, the resulting name includes the trailing blanks of the record, and the system cannot find the named file. Error messages are issued by the program attempting to locate the file.

See *z/OS XL C/C++ Programming Guide*, SC09-4765, for more information about specifying a list of environment variables using the _CEE_ENVFILE environment variable pointer.

B

Sample files provided with TCP/IP

This appendix lists the common sample files that are provided with the z/OS Communications Server product. You can find these samples at either of the following locations:

- `SYS1.SEZAINST`, an MVS PDS
- `/usr/lpp/tcpip/samples/`, an HFS directory

Sample files by component

Table B-1 lists the sample files that are provided with TCP/IP components.

Table B-1 Sample files provided with TCP/IP components

Resolver	
SEZAINST(RESOPROC)	JCL for the resolver proc
SEZAINST(RESSETUP)	Resolver setup file
General stack environment configuration	
SEZAINST(TCPIPROC)	JCL for the TCP/IP proc
SEZAINST(SAMPPROF)	PROFILE.TCPIP file for TCP/IP
SEZAINST(TCPDATA)	TCPIP.DATA file for TCP/IP
SEZAINST(IPNODES)	*hlq*.ETC.IPNODES file
SEZAINST(HOSTS)	HOSTS.LOCAL (or /etc/hosts) file
SEZAINST(CONVSYM)	symbol translator JCL job
SEZAINST(EZARACF)	SAF authorizations
SEZAINST(PROTO)	hlq.ETC.PROTO file
/usr/lpp/tcpip/samples/protocol	/etc/protocol file
SEZAINST(SERVICES)	ETC.SERVICES
/usr/lpp/tcpip/samples/services	/etc/services file
OMPROUTE	
SEZAINST(OMPROUTE)	JCL for OMPROUTE proc
SEZAINST(EZAORCFG)	OMPROUTE configuration file
SYSLOGD	
SEZAINST(SYSLOGD)	JCL for SYSLOGD proc
/usr/lpp/tcpip/samples/syslog.conf	SYSLOGD configuration file
/usr/lpp/tcpip/samples/rmoldlogs	Remove Old Logs script
TN3270E Telnet server	
SEZAINST(EZBTNPRC)	JCL for TN3270E Telnet server proc
SEZAINST(TNPROF)	TN3270 configuration file
SEZAINST(EZBTPUST)	3270 data stream USS table
SEZAINST(EZBTPSCS)	SCS data stream USS table
SEZAINST(EZBTPINT)	INTERPRET table
SEZAINST(EZBUSJCL)	JCL to assemble and link a USS table
SEZAINST(TNDBCSCN)	Double byte transform support
SEZAINST(VTAMLST)	VTAMLST definitions
SEZAINST(IVPLU)	VTAM sample APPL node for Telnet LUs
/usr/lpp/tcpip/samples/mvstn3270.mi2	mibs for TN3270

OTELNETD	
/usr/lpp/tcpip/samples/services	/etc/services file
/samples/inetd.conf	/etc/inetd.conf file
INETD	
/usr/lpp/tcpip/samples/services	/etc/services file
/samples/inetd.conf	/etc/inetd.conf file
FTP	
SEZAINST(FTPD)	JCL for FTPD proc
SEZAINST(FTCDATA)	FTP.DATA for the Client
SEZAINST(FTPSDATA)	FTP.DATA for the Server
TFTPD	
SEZAINST(TFTPD)	JCL for the TFTPD proc
IKED	
SEZAINST(IKED)	JCL for the IKED proc
/usr/lpp/tcpip/samples/iked.conf	Configuration file for IKED
SMTP	
SEZAINST(SMTPPROC)	JCL for SMTP proc
SEZAINST(SMTPCONF)	SMTP configuration file
SEZAINST(SMTPNOTE)	REXX source for SMTPNOTE
SEZAINST(SMTPEXIT)	SMTP exit controls “spam” mail
SEZAINST(CSSMTP)	JCL for CSSMTP proc
SEZAINST(CSSMTPCF)	CSSMTP configuration file
Sendmail	
SEZAINST(SENDMAIL)	JCL for sendmail proc
/usr/lpp/tcpip/samples/sendmail/cf/sample.cf	Configuration file for MTA and MUA
/usr/lpp/tcpip/samples/sendmail/cf/submit.cf	Configuration file for MUA specific
/usr/lpp/tcpip/samples/sendmail/cf/submit.mc	Input master configuration file
/usr/lpp/tcpip/samples/sendmail/sendmail.ps	Install and Ops Guide for sendmail 8.12
/usr/lpp/tcpip/samples/sendmail/cf/zOS.cf	z/OS-specific SSL for MTA
/usr/lpp/tcpip/samples/sendmail/ostype/zOS.m4	Describes the z/OS environment
/usr/lpp/tcpip/samples/sendmail/README.m4	Latest Instructions for sendmail
/usr/lpp/tcpip/samples/sendmail/feature/msp.m4	Special features msp file for sendmail
LPD/LPR	
SEZAINST(LPSPROC)	JCL for LPD server proc
SEZAINST(LPDDATA)	LPD.CONFIG configuration file
SEZAINST(EZAAE04S)	LPBANNER source file
SEZAINST(EZAAE04T)	LPBANNER translate table

DNS BIND9	
SEZAINST(NAMED9)	JCL for BIND 9 Proc
/usr/lpp/tcpip/samples/named.conf	Boot file for bind9
/usr/lpp/tcpip/samples/db.mycorp.v9	Sample version 9 file
/usr/lpp/tcpip/samples/db.34.37.9.v9	Sample version 9 file
/usr/lpp/tcpip/samples/db.loopback.v9	Sample version 9 file
/usr/lpp/tcpip/samples/slave.conf	Boot file for a subordinate server
/usr/lpp/tcpip/samples/caching.conf	Boot file for a caching server
ADNR (Automated Domain Name Registration)	
SEZAINST(ADNRCNF)	ADNR configuration statements
SEZAINST(ADNRPROC)	ADNR configuration statements
SNMP	
SEZAINST(OSNMPDPR)	JCL for osnmpd proc
SEZAINST(SNMPPROC)	JCL for snmpqe proc
SEZAINST(MIBDESC)	*hlq*.MIBDESC.DATA file
SEZAINST(EZASNTPR)	JCL for trap forwarder proc TRAPFWD
SEZAINST(TRAPFWD)	JCL for trap forwarder proc
SEZAINST(MSSNMP)	NLS Message data set for SNMPQE
/usr/lpp/tcpip/samples/mvstcpip.caps	SNMP agent capabilities statement
/usr/lpp/tcpip/samples/mvstcpip.mi2	IBM enterprise-specific mib
/usr/lpp/tcpip/samples/mvstn3270.mi2	TN3270 mib file
/usr/lpp/tcpip/samples/mibs.data	Textual names user mib file
/usr/lpp/tcpip/samples/snmpv2.conf	Configuration file for SNMPV2
/usr/lpp/tcpip/samples/snmpd.conf	Configuration file for SNMP
/usr/lpp/tcpip/samples/osnmpd.data	Control data file for SNMP
SNMPQE	
SEZAINST(SNMPPROC)	JCL for SNMPQE proc
REXEC	
SEZAINST(RXPROC)	JCL for REXECD proc
SEZAINST(RXUEXIT)	User exit for REXEC
UNIX REXECD	
/usr/lpp/tcpip/samples/syslog.conf	SYSLOGD configuration file
/usr/lpp/tcpip/samples/services	/etc/services file
/samples/inetd.conf	/etc/inetd.conf file
/usr/sbin/orexecd	location of program orexecd
/usr/lib/nls/msg/C/rexdmsg.cat	message catalog for rexecd

UNIX RSHD	
/usr/lpp/tcpip/samples/syslog.conf	SYSLOGD configuration file
/usr/lpp/tcpip/samples/services	/etc/services file
/samples/inetd.conf	/etc/inetd.conf file
/usr/sbin/orshd	Location of program orshd
/usr/sbin/ruserok	User verification
/usr/lib/nls/msg/C/rshdmsg.cat	Message catalog file for rshd
TIMED	
SEZAINST(TIMED)	JCL for TIMED proc
SNTPD	
SEZAINST(SNTPD)	JCL for SNTPD proc
PORTMAPPER	
SEZAINST(PORTPROC)	JCL for PORTMAPPER proc
SEZAINST(ETCRPC)	ETC.RPC control file
UNIX PORTMAPPER	
SEZAINST(OPORTPRC)	JCL for UNIX portmapper proc
RPCBIND	
SEZAINST(RPCBIND)	JCL for RPCBIND proc
/etc/services	Port reservation file for services
NCS INTERFACE	
SEZAINST(NRGLBD)	JCL for NCS Global Location Broker
SEZAINST(LLBD)	JCL for Local Location Broker
MISCSERV	
SEZAINST(MISCSERV)	JCL for Miscellaneous server proc
SEZAINST(MSMISCSR)	NLS Message data set for MISC Server
DCAS	
SEZAINST(DCAS)	JCL for DCAS proc
LBA (Load Balancing Advisor and Agent)	
SEZAINST(EZBLBADV)	JCL for Advisor proc
SEZAINST(EZBLBADC)	Advisor configuration file
SEZAINST(EZBLBAGE)	JCL for Agent proc
SEZAINST(EZBLBAGC)	Agent configuration file
SLAP	
SEZAINST(NSLAPM2)	JCL for SLAPM2 subagent proc
usr/lpp/tcpip/samples/slapm2.mi2	MIB file for NSLAPM2

RSVP Agent	
SEZAINST(RSVPD)	JCL for RSVPD proc
/usr/lpp/tcpip/samples/rsvpd.conf	Configuration file for RSVPD
TRMD	
SEZAINST(TRMD)	JCL for TRMD proc
Policy Agent	
SEZAINST(PAGENT)	JCL for Policy Agent proc
SEZAINST(EZAPOLPR)	JCL for Policy Agent Application Monitor Function proc
/usr/lpp/tcpip/samples/pagent.conf	Policy Agent configuration file
IPSec policy definitions	
/usr/lpp/tcpip/samples/pagent_CommonIPSec.conf	Configuration files: common IPSEc
/usr/lpp/tcpip/samples/pagent_IPSec.conf	Configuration files: stack-specific IPSec
AT-TLS policy definitions	
/usr/lpp/tcpip/samples/pagent_TTLS.conf	AT-TLS policy definitions
LDAP policy definitions	
/usr/lpp/tcpip/samples/pagent.ldif	LDAP top level directory structure
/usr/lpp/tcpip/samples/pagent_starter_QOS.ldif	Information file for QOS starter set
/usr/lpp/tcpip/samples/pagent_starter_IDS.ldif	Information file for IDS starter set
/usr/lpp/tcpip/samples/pagent_advanced_QOS.ldif	Information file for QOS advanced
/usr/lpp/tcpip/samples/pagent_advanced_IDS.ldif	Information file for IDS advanced
LDAP protocol version 3	
/usr/lpp/tcpip/samples/pagent_schema.ldif	V2 core and QOS object class
/usr/lpp/tcpip/samples/pagent_v3schema.ldif	V3 additions to V2 core
/usr/lpp/tcpip/samples/pagent_schema_updates.ldif	Updates to V2 core
/usr/lpp/tcpip/samples/pagent_idsschema.ldif	IDS schema file V3 for v1.4
/usr/lpp/tcpip/samples/pagent_qosschema.ldif	QOS schema file V3 for v1.5
/usr/lpp/tcpip/samples/pagent_r5idsschema.ldif	V3 IDS for v1.5
/usr/lpp/tcpip/samples/pagent_schema_r5updates.ldif	Updates for IDS v1.5
/usr/lpp/tcpip/samples/pagent_r6qosschema.ldif	QOS V3 for v1.6
/usr/lpp/tcpip/samples/pagent_schema_r6updates.ldif	Updates for V3 core for v1.6
Version 3 schema draft documents	
/usr/lpp/tcpip/samples/pagent_pcim.txt	Policy Core Information Model draft
/usr/lpp/tcpip/samples/pagent_core.txt	Policy Core LDAP draft for v1.2
/usr/lpp/tcpip/samples/pagent_cond.txt	Policy conditions draft for v1.2

C applications: policy performance monitoring	
/usr/lpp/tcpip/samples/pagent/README	Instructions for running the following pgms
/usr/lpp/tcpip/samples/pagent/pCollector.c	Test pgm: Policy API performance data
/usr/lpp/tcpip/samples/pagent/pCollector.h	Header file for pCollector pgm
/usr/lpp/tcpip/samples/pagent/pLogReader.c	Test pgm: reads policy performance log

C

Configuration files: TN3270E stand-alone scenario

This appendix contains the configuration files used for the TN3270E Telnet server stand-alone started task scenario.

This appendix includes the following sections:

- SC31 TN3270B Server PROC for TN3270 stand-alone scenario
- SC31 TN3270B Server profile for TN3270 stand-alone scenario
- SC31 TCPIPB stack PROC for TN3270 stand-alone scenario
- SC31 TCPIPB stack PROFILE for TN3270 stand-alone scenario
- SC31 OMPROUTE PROC for TN3270 stand-alone scenario
- SC31 OMPROUTE STDENV file for TN3270 stand-alone task scenario
- SC31 OMPROUTE CONFIG for TN3270 stand-alone scenario

SC31 TN3270B Server PROC for TN3270 stand-alone scenario

Figure C-1 lists the JCL procedure for the SC31 TN3270B server.

SYS1.PROCLIB(TN3270B)

```
//TN3270B   PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=TN3270&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//TN3270B  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
```

Figure C-1 SC31 TN3270B PROC JCL

We started the server task by issuing the following command:

```
S TN3270B,PROFILE=TELNB31A
```

SC31 TN3270B Server profile for TN3270 stand-alone scenario

The profile for the TN3270E Telnet server for this scenario is TELNB31A. The naming convention for the profiles is *XXXXyccs*:

- *XXXX* is TELN.
- *y* is B for the stack that was used for the examples in this book.
- *cc* is the &SYSCLONE value of this system.
- *s* is a letter representing the specific scenario (A, B, C, D).

Figure C-2 on page 417 and Figure C-3 on page 418 show this scenario.

```
 BROWSE    TCPIPB.TCPPARMS(TELNB31A) - 01.04          Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
; ===SC31============ TN3270 Server Profile for Standalone Task =======-
;                                                                        -
;   No SSL security.  No Sysplex Distribution in the stack.              -
;                                                                        -
; ---------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
; ---------------------------------------------------------------------
; These default device type settings will be used by all ports if no   -
; TELNETPARMS or PARMSGROUP is used to override the settings.          -
; They are logmode names shipped in ISTINCDT with the latest level of  -
; VTAM.                                                                -
; ---------------------------------------------------------------------
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
    ;
ENDTELNETGLOBALS
;
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
```

Figure C-2 (Part 1 of 2) SC31TN3270B profile (TELNB31A)

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC31BB01..SC31BB99
  ENDDEFAULTLUS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL SC*             ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION,5 ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
****************************** Bottom of Data ********************************
```

Figure C-3 (Part 2 of 2) SC31 TN3270B profile (TELNB30A)

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC31BB*nn*):

- *XXXX* is the &SYSNAME of the system.
- *y* is B for stack B that was used for all of the examples in this book.
- *z* is B for the basic port (23) and S for the secure port (992).
- *nn* is 01 through 99.

Because TN3270 maintains a master list of LU names in use across all ports, we could have used the same group of default LU names for all ports in our setup. We chose different pool names, one for basic and one for secure, to help us verify which port we are connected to in our display examples. Because there is only one port defined and in use in *this* scenario, all LU names will be shown as SC31BBnn.

SC31 TCPIPB stack PROC for TN3270 stand-alone scenario

Figure C-4 shows the JCL procedure for the stack is TCPIPB.

```
SYS1.PROCLIB(TCPIPB)
*********************************************************************************
//TCPIPB    PROC PARMS='CTRACE(CTIEZB00),IDS=00',
//             PROFILE=PROFB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE
//TCPIPB    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM=('&PARMS',
//         'ENVAR("_CEE_ENVFILE=DD:STDENV")')
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(NAMENV31)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure C-4 SC31 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB30A
```

SC31 TCPIPB stack PROFILE for TN3270 stand-alone scenario

The PROFILE for the stack is PROFB31, as shown in Figure C-5 through Figure C-9 on page 424.

```
 BROWSE    TCPIPB.TCPPARMS(PROFB31) - 01.06          Line 00000000 Col 001 080
 Command ===>                                              Scroll ===> CSR
******************************** Top of Data ***********************************
; added by bob
NETMONITOR SMFSERVICE
;
; TCPIP.PROFILE.TCPIP     for Standalone TN3270 and no Sysplex Dist.
; ==================
;
GLOBALCONFIG
   ECSALIMIT 0M              ; default = 0M
   POOLLIMIT 0M              ; default = 0M
   TCPIPSTATISTICS           ; default = notcpipstatistics
   XCFGRPID 21               ; subplex for TCPIPB on all lpars
   IQDVLANID 21
   SYSPLEXMONITOR DELAYJOIN   RECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   DATAGRAMFWD
   SYSPLEXROUTING
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
   DYNAMICXCF 10.1.7.21    255.255.255.0 8
;
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
;  RESTRICTLOWPORTS          ; default = unrestrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   NODELAYACKS               ; default = delayacks
;  TTLS                      ; default = NOTTLS
;
UDPCONFIG
;  RESTRICTLOWPORTS          ; default = unrestrictlowports
   UDPCHKSUM                 ; default = udpchksum
   UDPSENDBFRSIZE   65535    ; default = 64K
   UDPRCVBUFRSIZE   65535    ; default = 64K
 NOUDPQUEUELIMIT             ; default = udpqueuelimit
;
```

Figure C-5 (Part 1 of 5) SC31 TCPIPB profile (PROFB31)

```
 SMFCONFIG                    ;
    TYPE118 TCPIPSTATISTICS   ; default = notcpipstatistics
    TYPE119 TCPIPSTATISTICS   ; default = notcpipstatistics
          NOTCPINIT           ; default = notcpinit
          NOTCPTERM           ; default = notcpterm
            FTPCLIENT         ; default = noftpclient
            TN3270CLIENT      ; default = notn3270client
            IFSTATISTICS      ; default = noifstatistics
            PORTSTATISTICS    ; default = noportstatistics
            TCPSTACK          ; default = notcpstack
          NOUDPTERM           ; default = noudpterm
;
 AUTOLOG 5

   OMPB
   FTPDB
   INETDB
;  LPSERVEB
;  RXSERVE
   SENDMAIL
   SNMPDB
   SNMPOSAB
   SNMPQEB
   TRAPFWDB
   RPCBIND JOBNAME RPCBIND1

 ENDAUTOLOG
;
;INTERFACE OSA20x0I DEFINE IPAQENET (OSA-E) PORTNAME OSA20x0
;TRL MAJ NODE: OSA2080,OSA20A0,OSA20C0,AND OSA20E0
;
INTERFACE OSA2080I
DEFINE IPAQENET
PORTNAME OSA2080
IPADDR 10.1.2.21/24
MTU 1492
VLANID 10
VMAC
;
INTERFACE OSA20A0I
DEFINE IPAQENET
PORTNAME OSA20A0
IPADDR 10.1.2.22/24
MTU 1492
VLANID 10
VMAC
;
INTERFACE OSA20C0I
DEFINE IPAQENET
PORTNAME OSA20C0
IPADDR 10.1.3.21/24
MTU 1492
VLANID 11
VMAC
;
```

Figure C-6 (Part 2 of 5) SC31 TCPIPB profile (PROFB31)

```
INTERFACE OSA2OE0I
DEFINE IPAQENET
PORTNAME OSA2OE0
IPADDR 10.1.3.22/24
MTU 1492
VLANID 11
VMAC
;
;HiperSockets definitions
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4L    IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5L    IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6L    IPAQIDIO      IUTIQDF6
;
; Static VIPA definitions                                      -
 DEVICE VIPA1      VIRTUAL 0
 LINK   VIPA1L     VIRTUAL 0   VIPA1
;
 VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set aside some addresses for use with BIND and SIOCSVIPA IOCTL  -
  ;               (10.1.9.21 thru 10.1.9.24)                       -
  ;-------------------------------------------------------------------
   VIPARANGE define move nondisrupt 255.255.255.0  10.1.9.0

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.25    ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25     PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21    ;General
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                         10.1.8.21     PORT 992 20 21 23
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22    ;Admin
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.22     PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
```

Figure C-7 (Part 3 of 5) SC31 TCPIPB profile (PROFB31)

```
 ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23  ; Payrol
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.24  ; EXTRAS
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.24   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;      Distribute to 10.1.1.10    via IP routing   (   viparoute)  -
  ;      Distribute to 10.1.1.20    via normal XCF   (no viparoute)  -
  ;-------------------------------------------------------------------
   VIPAROUTE      DEFINE 10.1.7.11    10.1.1.10   ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.1.7.21    10.1.1.20   ; sc31's static vipa

 ENDVIPADYNAMIC
;
HOME
   10.1.1.20     VIPA1L
   10.1.4.21     IUTIQDF4L
   10.1.5.21     IUTIQDF5L
   10.1.6.21     IUTIQDF6L
;
  PRIMARYINTERFACE VIPA1L
;
PORT
     7 UDP MISCSRVB            ; Miscellaneous Server - echo
     7 TCP MISCSRVB            ; Miscellaneous Server - echo
     9 UDP MISCSRVB            ; Miscellaneous Server - discard
     9 TCP MISCSRVB            ; Miscellaneous Server - discard
    19 UDP MISCSRVB            ; Miscellaneous Server - chargen
    19 TCP MISCSRVB            ; Miscellaneous Server - chargen
    20 TCP OMVS                ; FTP Server
    21 TCP FTPDB1                         ; control port
;   21 TCP FTPDB1  bind 10.1.9.11         ; control port
;   23 TCP OMVS    BIND 10.1.9.23         ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG              ;   BIND 10.1.8.21
;   25 TCP SMTPB                          ; SMTP Server
;   25 TCP SENDMAIL                       ; Portmap Server
    53 TCP OMVS                           ; Domain Name Server
    53 UDP OMVS                           ; Domain Name Server
   110 TCP INETDB1                         ; Portmap Server
   111 TCP RPCBIND1                       ; Portmap Server
   111 UDP RPCBIND1                       ; Portmap Server
;  123 UDP SNTPD              ; Simple Network Time Protocol Server
;  135 UDP LLBD               ; NCS Location Broker
```

Figure C-8 (Part 4 of 5) SC31 TCPIPB profile (PROFB31)

```
 161 UDP SNMPDB                       ; SNMP Agent 162 UDP SNMPQEB
; SNMP Query Engine
;  389 TCP LDAPSRV             ; LDAP Server
;  443 TCP HTTPS               ; http protocol over TLS/SSL
;  443 UDP HTTPS               ; http protocol over TLS/SSL
;  512 TCP OMVS BIND 10.1.9.22           ; UNIX REXECD D-VIPA
;  514 TCP OMVS BIND 10.1.9.22           ; UNIX RSHD   D-VIPA
   512 TCP RXSERVEB                      ; TSO  REXECD
   514 TCP RXSERVEB                      ; TSO  RSHD
   515 TCP LPSERVEB                      ; LPD Server
   520 UDP OMPB    NOAUTOLOG             ; OMPROUTE
;  580 UDP NCPROUT             ; NCPROUTE Server
   722 TCP CS*                 ; For LPR printing by users CSxx
   723 TCP CS*                 ; For LPR printing by users CSxx
   724 TCP CS*                 ; For LPR printing by users CSxx
   725 TCP CS*                 ; For LPR printing by users CSxx
   726 TCP CS*                 ; For LPR printing by users CSxx
   727 TCP CS*                 ; For LPR printing by users CSxx
   728 TCP CS*                 ; For LPR printing by users CSxx
   729 TCP CS*                 ; For LPR printing by users CSxx
   730 TCP CS*                 ; For LPR printing by users CSxx
   731 TCP CS*                 ; For LPR printing by users CSxx
   750 TCP MVSKERBB                      ; Kerberos
   750 UDP MVSKERBB                      ; Kerberos
   751 TCP ADM@SRVB                      ; Kerberos Admin Server
   751 UDP ADM@SRVB                      ; Kerberos Admin Server
   992 TCP TN3270B NOAUTOLOG   ; Secure TN3270 via internal SSL
  4992 TCP TN3270B NOAUTOLOG   ; Secure TN3270 via AT-TLS
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161

ITRACE OFF

START OSA2080I
START OSA20A0I
START OSA20C0I
START OSA20E0I
START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
;
```

Figure C-9 (Part 5 of 5) SC31 TCPIPB profile (PROFB31)

SC31 OMPROUTE PROC for TN3270 stand-alone scenario

Figure C-10 shows the JCL procedure for OMPROUTE.

```
SYS1.PROCLIB(OMPB)
***************************** Top of Data ********************
//OMPB   PROC STDENV=OMPENB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=OM,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
***************************** Bottom of Data ******************
```

Figure C-10 SC31 OMPROUTE PROC (OMPB)

SC31 OMPROUTE STDENV file for TN3270 stand-alone task scenario

Figure C-11 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIP.SC31.STDENV(OMPENB31) - 01.04          Line 00000000 Col 001 080
 Command ===>                                              Scroll ===> CSR
******************************** Top of Data *********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB31)'
OMPROUTE_DEBUG_FILE=/tmp/syslog/debugb31
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ******************************
```

Figure C-11 SC31 OMPROUTE STDENV file (OMPENB31)

SC31 OMPROUTE CONFIG for TN3270 stand-alone scenario

Figure C-12 through Figure C-14 on page 427 show the config for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB31) - 01.00            Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
Area Area_Number=0.0.0.2
 Stub_Area=YES
 Authentication_type=None;
;
;New Parameter OSPF
OSPF
 RouterID=10.1.1.20
 Comparison=Type2
 Demand_Circuit=YES;
Global_Options
 Ignore_Undefined_Interfaces=YES
;
Routesa_Config Enabled=Yes Community="jOs9m2ap" Agent=161;
;
; Static vipa
ospf_interface ip_address=10.1.1.20
            name=VIPA1L
            subnet_mask=255.255.255.0
            Advertise_VIPA_Routes=HOST_ONLY
            attaches_to_area=0.0.0.2
            cost0=10
            mtu=65535;
;
; OSA Qdio 10.1.2.x
ospf_interface ip_address=10.1.2.*
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=0
            attaches_to_area=0.0.0.2
            cost0=100
            mtu=1492;
; OSA Qdio 10.1.3.x
ospf_interface ip_address=10.1.3.*
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=0
            attaches_to_area=0.0.0.2
            cost0=90
            mtu=1492;
;
; Hipersockets 10.1.4.x
ospf_interface ip_address=10.1.4.*
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=1
            attaches_to_area=0.0.0.2
            cost0=80
            mtu=8192;
```

Figure C-12 (Part 1 of 3) SC31 OMPROUTE config (OMPB31)

```
; Hipersockets 10.1.5.x
ospf_interface ip_address=10.1.5.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=1
           attaches_to_area=0.0.0.2
           cost0=80
           mtu=8192;
;
; Dynamic vipa VIPADEFINE
ospf_interface ip_address=10.1.8.*
           subnet_mask=255.255.255.0
           Advertise_VIPA_Routes=HOST_ONLY
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=65535;
;
; Dynamic vipa VIPARANGE
ospf_interface ip_address=10.1.9.*
           subnet_mask=255.255.255.0
           Advertise_VIPA_Routes=HOST_ONLY
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=65535;
```

Figure C-13 (Part 2 of 3) SC31 OMPROUTE config (OMPB31)

```
;
; ************************************************************
; *XCF Interfaces
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to just code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
INTERFACE
     IP_Address=10.1.7.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure C-14 (Part 3 of 3) SC30 OMPROUTE Config (OMPB31)

D

Multiple TN3270E Telnet servers and sysplex distribution using the LUNS and LUNR scenario

This appendix contains the configuration files that are used for the multiple TN3270E Telnet servers with the sysplex distribution (SD) scenario. It includes the following sections:

- SC30 files for LUNS and LUNR scenario
- SC31 files for LUNS and LUNR scenario

SC30 files for LUNS and LUNR scenario

This section includes the following topics:

- SC30 TN3270A Server PROC for LUNS and LUNR scenario
- SC30 TN3270A Server PROFILE for LUNS and LUNR scenario
- SC30 TNLUNS30 backup LUNS PROC for LUNS and LUNR scenario
- SC30 TNLUNS30 PROFILE for LUNS and LUNR scenario
- SC30 TCPIPA stack PROC for LUNS and LUNR scenario
- SC30 TCPIPA stack PROFILE for LUNS and LUNR scenario
- SC30 OMPROUTE PROC for LUNS and LUNR scenario
- SC30 OMPROUTE STDENV file for LUNS and LUNR scenario
- SC30 OMPROUTE CONFIG for LUNS and LUNR scenario

SC30 TN3270A Server PROC for LUNS and LUNR scenario

Figure D-1 shows the PROC JCL for the TN3270E Telnet server.

```
 BROWSE    SYS1.PROCLIB(TN3270A) - 01.02             Line 00000000 Co
 Command ===>                                                Scroll
******************************** Top of Data ***********************
//TN3270A    PROC PARMS='CTRACE(CTIEZBTN)',
//            PROFILE=TN3270&SYSCLONE.,TCPDATA=DATAA&SYSCLONE.
//TN3270A  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(&TCPDATA.)
******************************** Bottom of Data ********************
```

Figure D-1 SC30 TN3270A PROC JCL

We started the server task for this scenario by issuing the following command:

```
S TN3270A,PROFILE=TELNA30C
```

SC30 TN3270A Server PROFILE for LUNS and LUNR scenario

The profile for the TN3270E Telnet server for this scenario is TELNA30C. The naming convention for the profiles is *XXXXyccs*:

- *XXXX* is TELN.
- *y* is B for the stack that was used for the examples in this book.
- *cc* is the &SYSCLONE value of this system.
- *s* is a letter representing the specific scenario, (A, B, C, D).

Figure D-2 through Figure D-5 on page 434 show this scenario.

```
 BROWSE    TCPIPA.TCPPARMS(TELNA30C) - 01.04          Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
; ===SC30============ TN3270 Server Profile for SYSPLEX Task ==========-
;                                                                       -
;    SSL security.  Sysplex Distribution.  LUNS/LUNR.                   -
;                                                                       -
; ---------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPA
    TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
    XCFGROUP
        JOIN XCFMONITOR 60
        RECOVERYTIMEOUT 60
        CONNECTTIMEOUT 60
    ENDXCFGROUP
    TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
;   XCFGROUP
;       JOIN XCFMONITOR 60
;       LUNS 10.1.1.10 4444
;       BACKUP
;       RANK 99
;       RECOVERYTIMEOUT 60
;       CONNECTTIMEOUT 60
;   ENDXCFGROUP
; ---------------------------------------------------------------------
; These default device type settings will be used by all ports if no   -
; TELNETPARMS or PARMSGROUP is used to override the settings.          -
; They are logmode names shipped in ISTINCDT with the latest level of  -
; VTAM.                                                                -
; ---------------------------------------------------------------------
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
    ;
ENDTELNETGLOBALS
;
```

Figure D-2 (Part 1 of 4) SC30 TN3270A profile (TELNA30C)

```
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 23

  SDEFAULTLUS
      SHLU01..SHLU04
  ENDSDEFAULTLUS

  SLUGROUP
      SHRGRP99  SH99LU01..SH99LU04
  ENDSLUGROUP

  LUMAP SHRGRP99 10.1.100.221

  MONITORGROUP SNAONLY
    AVERAGE
    BUCKETS
    NODYNAMICDR
    NOINCLUDEIP
    AVGSAMPPERIOD 120
    AVGSAMPMULTIPLIER 5
    BOUNDARY1 1
    BOUNDARY2 2
    BOUNDARY3 3
    BOUNDARY4 4
  ENDMONITORGROUP

  DESTIPGROUP ALLUSERS    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  MONITORMAP SNAONLY    DESTIPGRP,ALLUSERS

  DEFAULTAPPL SC30TS        ; All users go to TSO
  ALLOWAPPL SC*             ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
```

Figure D-3 (Part 2 of 4) SC30 TN3270A profile (TELNA30C)

```
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING SAF TCPIP/SharedRing1       ;keyring shared by servers
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
ENDTELNETPARMS
;
BEGINVTAM
  PORT 992
  DEFAULTLUS
      SC30BS01..SC30BS99
  ENDDEFAULTLUS
; ------------------------------------------------------------------
; This NOSSL        group is mapped to use no SSL security.          -
; NOLUSESSIONPEND = Terminate connection upon a logoff
; ------------------------------------------------------------------
  PARMSGROUP NOSSL
   NOLUSESSIONPEND
     CONNTYPE BASIC  ; support non-secure, overrides telnetparms
  ENDPARMSGROUP
; ------------------------------------------------------------------
; The SSLPLAIN     group is mapped to use SSL security               -
;                  with no Client Authentication required           -
; LUSESSIONPEND = Force a requeue back to the USS table upon logoff -
; ------------------------------------------------------------------
  PARMSGROUP SSLPLAIN
     LUSESSIONPEND
     CONNTYPE SECURE ; says plain SSL, no client auth specified
  ENDPARMSGROUP      ; and negotiate all available encryption algorithms
; ------------------------------------------------------------------
; The SSLCERTS     group is mapped to use SSL security               -
;                  and to require Client Authentication (certificates) -
; NOLUSESSIONPEND = Terminate connection upon a logoff from the appl -
; ------------------------------------------------------------------
  PARMSGROUP SSLCERTS
   NOLUSESSIONPEND
     CONNTYPE SECURE       ; Support SSL
     CLIENTAUTH SSLCERT    ; Client Certificate required
     ENCRYPT SSL_DES_SHA   ; use these only, do not consider any others
             SSL_3DES_SHA
     ENDENCRYPT
  ENDPARMSGROUP
; ------------------------------------------------------------------
; The UP2USER      group is mapped to use ANY security (user's choice) -
;                  with no Client Authentication (no certificates)     -
; LUSESSIONPEND = Force a requeue back to the defaultappl upon logoff  -
; ------------------------------------------------------------------
  PARMSGROUP UP2USER
     LUSESSIONPEND
     CONNTYPE ANY          ; Whatever User wants to do
  ENDPARMSGROUP
```

Figure D-4 (Part 3 of 4) SC30 TN3270A profile (TELNA30C)

```
 MONITORGROUP SNAANDIP
    AVERAGE
    BUCKETS
    DYNAMICDR
    INCLUDEIP
    AVGSAMPPERIOD 120
    AVGSAMPMULTIPLIER 5
    BOUNDARY1 1
    BOUNDARY2 2
    BOUNDARY3 3
    BOUNDARY4 4
  ENDMONITORGROUP

  DESTIPGROUP GENERALUSER 10.1.8.21     ENDDESTIPGROUP
  DESTIPGROUP ADMIN       10.1.8.22     ENDDESTIPGROUP
  DESTIPGROUP PAYROLL     10.1.8.23     ENDDESTIPGROUP
  DESTIPGROUP SHIPPING    10.1.1.10     ENDDESTIPGROUP
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30TS    DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLPO  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER         DESTIPGRP,SHIPPING
  USSTCP USSTABEE,USSSNAEE DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE

  PARMSGROUP DEBUGIT  DEBUG TRACE JOBLOG ENDPARMSGROUP
  IPGROUP DEBUGIPGRP 10.1.100.221
                     10.1.100.222
                     10.1.100.223
                     10.1.100.224
  ENDIPGROUP
PARMSMAP DEBUGIT  IPGRP,DEBUGIPGRP
MONITORMAP SNAANDIP   DESTIPGRP,GENERALUSER
;-------------------------------------------------------------------
 ; There is no DEFAULTAPPL nor USSTCB coded as a default catch all -
 ; So, if any user connects using any other IP address than the    -
 ; four defined by the DESTIPGROUPs above, the Network Solicitor   -
 ; prompt panel will be displayed to that user, in BASIC mode.     -
 ;-------------------------------------------------------------------
  ALLOWAPPL SC3*            ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
****************************** Bottom of Data ********************************
```

Figure D-5 (Part 4 of 4) SC30 TN3270A profile (TELNA30C)

SC30 TNLUNS30 backup LUNS PROC for LUNS and LUNR scenario

Figure D-6 shows the PROC JCL for the backup LUNS TNLUNS30.

```
//TNLUNS30  PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=LUNS&SYSCLONE.,TCPDATA=DATAA&SYSCLONE.
//TNLUNS   EXEC PGM=EZBTNINI,REGION=OM,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(&TCPDATA.)
```

Figure D-6 SC30 TNLUNS30 PROC JCL

We started the TNLUNS30 started task for this scenario by issuing the following command:

```
S TNLUNS30
```

SC30 TNLUNS30 PROFILE for LUNS and LUNR scenario

Figure D-7 shows the profile for the backup LUNS is LUNS30.

```
 BROWSE    TCPIPA.TCPPARMS(LUNS30) - 01.23            Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
******************************** Top of Data ********************************
; ===SC30============ backup LUNS Pofile for LUNS/LUNR ================-
;                                                                          -
;    SSL security.  Sysplex Distribution. LUNS/LUNR                        -
;                                                                          -
; -------------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPA
    XCFGROUP
        JOIN XCFMONITOR 60
        LUNS 10.1.1.10 4444
        BACKUP
        RANK 99
    ENDXCFGROUP
    TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
ENDTELNETGLOBALS
****************************** Bottom of Data ******************************
```

Figure D-7 SC30 backup LUNS profile LUNS30

SC30 TCPIPA stack PROC for LUNS and LUNR scenario

Figure D-8 shows that the PROC JCL for the stack is TCPIPA.

```
//TCPIPA    PROC PARMS='CTRACE(CTIEZB00),IDS=00',
//             PROFILE=PROFA&SYSCLONE,TCPDATA=DATAA&SYSCLONE
//TCPIPA    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM=('&PARMS',
//          'ENVAR("RESOLVER_CONFIG=//''TCPIPA.TCPPARMS(&TCPDATA)''")')
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPA.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure D-8 SC30 TCPIPA PROC JCL

We started the TCPIPA started task for this scenario by issuing the following command:

```
S TCPIPA
```

SC30 TCPIPA stack PROFILE for LUNS and LUNR scenario

The PROFILE for the stack is PROFA30. See Figure D-9 through Figure D-13.

```
 BROWSE    TCPIPA.TCPPARMS(PROFA30) - 01.10          Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
;  This profile is for V1R11 Redbook -- Dynamic Routing
ARPAGE 20
;
GLOBALCONFIG NOTCPIPSTATISTICS IQDMULTIWRITE ZIIP IQDIOMULTIWRITE
GLOBALCONFIG SEGMENTATIONOFFLOAD
GLOBALCONFIG EXPLICITBINDPORTRANGE 7000 3
GLOBALCONFIG XCFGRPID 21 IQDVLANID 21
GLOBALCONFIG SYSPLEXMONITOR DELAYJOIN   RECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   DATAGRAMFWD
   SYSPLEXROUTING
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
   DYNAMICXCF 10.1.7.11    255.255.255.0 8
;
NETMONITOR SMFSERVICE
;
SOMAXCONN 10
;
TCPCONFIG TCPSENDBFRSIZE 256K TCPRCVBUFRSIZE 256K SENDGARBAGE FALSE
TCPCONFIG TCPMAXRCVBUFRSIZE 512K
TCPCONFIG UNRESTRICTLOWPORTS
;TCPCONFIG TTLS
;
UDPCONFIG UNRESTRICTLOWPORTS UDPSENDBFRSIZE 65535 UDPRCVBUFRSIZE 65535
UDPCONFIG NOUDPQUEUELIMIT
;
;INTERFACE OSA20x0I DEFINE IPAQENET (OSA-E) PORTNAME OSA20x0
;TRL MAJ NODE: OSA2080,OSA20A0,OSA20C0,AND OSA20E0
;
INTERFACE OSA2080I
DEFINE IPAQENET
PORTNAME OSA2080
IPADDR 10.1.2.11/24
MTU 1492
VLANID 10
VMAC
;
INTERFACE OSA2081I
DEFINE IPAQENET
PORTNAME OSA2081
IPADDR 10.1.2.14/24
MTU 1492
VLANID 10
VMAC
;
```

Figure D-9 (Part 1 of 5) SC30 TCPIPA profile (PROFA30)

```
INTERFACE OSA20A0I
DEFINE IPAQENET
PORTNAME OSA20A0
IPADDR 10.1.2.12/24
MTU 1492
VLANID 10
VMAC
;
INTERFACE OSA20C0I
DEFINE IPAQENET
PORTNAME OSA20C0
IPADDR 10.1.3.11/24
MTU 1492
VLANID 11
VMAC
;
INTERFACE OSA20E0I
DEFINE IPAQENET
PORTNAME OSA20E0
IPADDR 10.1.3.12/24
MTU 1492
VLANID 11
VMAC
;
;HIPERSOCKETS DEFINITIONS
DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4
DEVICE IUTIQDF5   MPCIPA
LINK   IUTIQDF5L  IPAQIDIO    IUTIQDF5
DEVICE IUTIQDF6   MPCIPA
LINK   IUTIQDF6L  IPAQIDIO    IUTIQDF6
;
;STATIC VIPA DEFINITIONS
DEVICE VIPA1      VIRTUAL 0
LINK   VIPA1L     VIRTUAL 0   VIPA1
DEVICE VIPA2      VIRTUAL 0
LINK   VIPA2L     VIRTUAL 0   VIPA2
;
;DYNAMIC VIPA DEFINITIONS
VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.25    ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25     PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21    ;General
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                         10.1.8.21     PORT 992 20 21 23
                  DESTIP 10.1.7.11
                         10.1.7.21
```

Figure D-10 (Part 2 of 5) SC30 TCPIPA profile (PROFA30)

```
;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22    ;Admin
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.22     PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23  ; Payrol
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.24  ; EXTRAS
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.24   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
;-------------------------------------------------------------------
  ;      Distribute to 10.1.1.10    via IP routing   (   viparoute)  -
  ;      Distribute to 10.1.1.20    via normal XCF   (no viparoute)  -
;-------------------------------------------------------------------
   ;VIPAROUTE      DEFINE 10.1.7.11    10.1.1.10   ; sc30's static vipa
    VIPAROUTE      DEFINE 10.1.7.21    10.1.1.20   ; sc31's static vipa
ENDVIPADYNAMIC
;
HOME
   10.1.1.10     VIPA1L
   10.1.2.10     VIPA2L
   10.1.4.11     IUTIQDF4L
   10.1.5.11     IUTIQDF5L
   10.1.6.11     IUTIQDF6L
;
 PRIMARYINTERFACE VIPA1L
;
AUTOLOG 5
   FTPDA  JOBNAME FTPDA1
   OMPA
   IOASRV
;  LPSERVE                 ; LPD Server
;  NAMED                   ; Domain Name Server
;  NCPROUT                 ; NCPROUTE Server
;  OROUTED                 ; OROUTED Server
;  OSNMPD                  ; SNMP Agent Server
;  PORTMAP JOBNAME PORTMAP  ; zUNIX Portmap Server (SUN 4.0)
;  REXECD                  ; Remote Execution Server
;  SMTP                    ; SMTP Server
;  SNMPQE                  ; SNMP Client
;  TCPIPX25                ; X25 Server
ENDAUTOLOG
```

Figure D-11 (Part 3 of 5) SC30 TCPIPA profile (PROFA30)

```
 ;
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 OMVS NOAUTOLOG      ; FTP Server
    21 TCP FTPDA1 BIND 10.1.1.10 ;Control port
;   23 TCP INTCLIEN            ; MVS Telnet Server
; 1023 TCP INTCLIEN            ; MVS Telnet Server
;   23 TCP OMVS BIND 10.1.9.11 ; OE Telnet Server
    23 TCP TN3270A             ; 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
;  111 UDP PORTMAP             ; Portmap Server
;  123 UDP SNTPD               ; Simple Network Time Protocol Server
;  135 UDP LLBD                ; NCS Location Broker
;  161 UDP OSNMPD              ; SNMP Agent
;  162 UDP SNMPQE              ; SNMP Query Engine
;  389 TCP LDAPSRV             ; LDAP Server
;  443 TCPmHTTPS               ; http protocol over TLS/SSL
;  443 UDP HTTPS               ; http protocol over TLS/SSL
   500 UDP IKED                ; @ADI
;  512 TCP OMVS BIND 9.12.4.212   ; Remote Execution Server
;  514 TCP OMVS BIND 9.12.4.212   ; Remote Execution Server
   514 UDP OMVS                ; Remote Execution Server
;  515 TCP LPSERVE             ; LPD Server
   520 UDP OMPROUTE NOAUTOLOG  ; OMPROUTE RIPV2 port
   521 UDP OMPROUTE NOAUTOLOG  ; OMPROUTE RIPV2 port
;  580 UDP NCPROUT             ; NCPROUTE Server
;  623 TCP INTCLIEN            ; Telnet 3270 Server
;  750 TCP MVSKERB             ; Kerberos
;  750 UDP MVSKERB             ; Kerberos
;  751 TCP ADM@SRV             ; Kerberos Admin Server
;  751 UDP ADM@SRV             ; Kerberos Admin Server
; 1933 TCP ILMTSRVR            ; IBM LM MT Agent
; 1934 TCP ILMTSRVR            ; IBM LM Appl Agent
  2000 TCP IOASRV
; 3000 TCP CICSTCP             ; CICS Socket
; 3389 TCP MSYSLDAP            ; LDAP Server for Msys
  4500 UDP IKED                ; @ADI
PORTRANGE 10000 2000 TCP OMVS  ; TCP 10000 - 11999
PORTRANGE 10000 2000 UDP OMVS  ; UDP 10000 - 11999
;
SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
;
SMFCONFIG TYPE119 TCPINIT TCPTERM TCPIPSTATISTICS TN3270CLIENT
          FTPCLIENT
```

Figure D-12 (Part 4 of 5) SC30 TCPIPA profile (PROFA30)

```
;
START OSA2080I
START OSA2081I
START OSA20A0I
START OSA20C0I
START OSA20E0I
START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
******************************** Bottom of Data ********************************
```

Figure D-13 (Part 5 of 5) SC30 TCPIPA profile (PROFA30)

SC30 OMPROUTE PROC for LUNS and LUNR scenario

Figure D-14 shows the PROC JCL for OMPROUTE.

```
***************************** Top of Data ********************
//*
//OMPA   PROC STDENV=OMPENA&SYSCLONE
//OMPA   EXEC PGM=OMPROUTE,REGION=OM,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//             'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPA"',
//             '"_CEE_ENVFILE=DD:STDENV")/-d1')
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//OMPCFG   DD DSN=TCPIPA.TCPPARMS(OMPA&SYSCLONE.),DISP=SHR
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
***************************** Bottom of Data ******************
```

Figure D-14 SC30 OMPROUTE PROC (OMPA)

SC30 OMPROUTE STDENV file for LUNS and LUNR scenario

Figure D-15 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIP.SC30.STDENV(OMPENA30) - 01.04          Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data *********************************
; sysclone not allowed on OMPROUTE_FILE
; sysclone IS allowed on RESOLVER_CONFIG
RESOLVER_CONFIG=//'TCPIPA.TCPPARMS(DATAA&SYSCLONE.)'
OMPROUTE_DEBUG_FILE=/tmp/syslog/debuga30
OMPROUTE_DEBUG_FILE_CONTROL=10000,5
******************************** Bottom of Data ******************************
```

Figure D-15 SC30 OMPROUTE STDENV file (OMPENA30)

SC30 OMPROUTE CONFIG for LUNS and LUNR scenario

Figure D-16 and Figure D-17 on page 443 show the config for OMPROUTE.

```
 BROWSE    TCPIPA.TCPPARMS(OMPA30) - 01.00            Line 00000000 Col 001 080
 Command ===>                                           Scroll ===> CSR
******************************** Top of Data ********************************
Area Area_Number=0.0.0.2
 Stub_Area=YES
 Authentication_type=None;
;
;New Parameter OSPF
OSPF
 RouterID=10.1.1.10
 Comparison=Type2
 DR_Max_Adj_Attempt = 10
 Demand_Circuit=YES;
Global_Options
 Ignore_Undefined_Interfaces=YES ;
; Static vipa
ospf_interface ip_address=10.1.1.10
            name=VIPA1L
            subnet_mask=255.255.255.0
            Advertise_VIPA_Routes=HOST_ONLY
            attaches_to_area=0.0.0.2
            cost0=10
            mtu=65535;
;
ospf_interface ip_address=10.1.2.10
            name=VIPA2L
            subnet_mask=255.255.255.0
            Advertise_VIPA_Routes=HOST_ONLY
            attaches_to_area=0.0.0.2
            cost0=10
            mtu=65535;
;
; OSA Qdio 10.1.2.x
ospf_interface ip_address=10.1.2.*
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=0
            attaches_to_area=0.0.0.2
            cost0=100
            mtu=1492;
;
; OSA Qdio 10.1.3.x
ospf_interface ip_address=10.1.3.*
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=0
            attaches_to_area=0.0.0.2
            cost0=100
            mtu=1492;
;
; Hipersockets 10.1.4.x
ospf_interface ip_address=10.1.4.*
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=1
            attaches_to_area=0.0.0.2
            cost0=80
            mtu=8192;
;
```

Figure D-16 (Part 1 of 2) SC30 OMPROUTE Config (OMPA30)

```
; Hipersockets 10.1.5.x
ospf_interface ip_address=10.1.5.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=1
           attaches_to_area=0.0.0.2
           cost0=80
           mtu=8192;
;
; Hipersockets 10.1.6.x
ospf_interface ip_address=10.1.6.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=1
           attaches_to_area=0.0.0.2
           cost0=190
           mtu=8192;
;
; Dynamic vipa VIPADEFINE
ospf_interface ip_address=10.1.8.*
           subnet_mask=255.255.255.0
           Advertise_VIPA_Routes=HOST_ONLY
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=65535;
;
; Dynamic vipa VIPARANGE
ospf_interface ip_address=10.1.9.*
           subnet_mask=255.255.255.0
           attaches_to_area=0.0.0.2
           Advertise_VIPA_Routes=HOST_ONLY
           cost0=10
           mtu=65535;
;Dynamic XCF
interface  ip_address=10.1.7.*
           subnet_mask=255.255.255.0
           mtu=65535;
```

Figure D-17 (Part 2 of 2) SC30 OMPROUTE Config (OMPA30)

SC31 files for LUNS and LUNR scenario

This section includes the following topics:

- SC31 TN3270B Server PROC for LUNS and LUNR scenario
- SC31 TN3270B Server PROFILE for LUNS and LUNR scenario
- SC31 TNLUNS31 primary LUNS PROC for LUNS and LUNR scenario
- SC31 TNLUNS31 PROFILE for LUNS and LUNR scenario
- SC31 TCPIPB stack PROC for LUNS and LUNR scenario
- SC31 TCPIPB stack PROFILE for LUNS and LUNR scenario
- SC31 OMPROUTE PROC for LUNS and LUNR scenario
- SC31 OMPROUTE STDENV file for LUNS and LUNR scenario
- SC31 OMPROUTE CONFIG for LUNS and LUNR scenario

SC31 TN3270B Server PROC for LUNS and LUNR scenario

Figure D-18 shows the PROC JCL for the TN3270E Telnet server.

```
 BROWSE    SYS1.PROCLIB(TN3270B) - 01.02             Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ***********************************
//TN3270B   PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=TN3270&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//TN3270B  EXEC PGM=EZBTNINI,REGION=OM,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
```

Figure D-18 SC31 TN3270B PROC JCL

We started the server task for this scenario by issuing the following command:

```
S TN3270B,PROFILE=TELNB31C
```

SC31 TN3270B Server PROFILE for LUNS and LUNR scenario

The profile for the TN3270E Telnet server for this scenario is TELNB31C. The naming convention for the profiles is *XXXXyccs*:

- *XXXX* is TELN.
- *y* is B for the stack that was used for the examples in this book.
- *cc* is the &SYSCLONE value of this system.
- *s* is a letter representing the specific scenario, (A, B, C, D).

Figure D-19 through Figure D-22 on page 448 show this scenario.

```
 BROWSE    TCPIPB.TCPPARMS(TELNB31C) - 01.03          Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
; ===SC31============ TN3270 Server Profile for Sysplex Task ==========-
;                                                                       -
;    SSL security.  Sysplex Distribution. LUNS/LUNR.                    -
;                                                                       -
; ---------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
    TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
    XCFGROUP
       JOIN XCFMONITOR 60
       RECOVERYTIMEOUT 60
       CONNECTTIMEOUT 60
   ENDXCFGROUP
; ---------------------------------------------------------------------
; These default device type settings will be used by all ports if no   -
; TELNETPARMS or PARMSGROUP is used to override the settings.          -
; They are logmode names shipped in ISTINCDT with the latest level of  -
; VTAM.                                                                -
; ---------------------------------------------------------------------
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
    ;
ENDTELNETGLOBALS
;
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
```

Figure D-19 (Part 1 of 4) SC31 TN3270B profile (TELNB31C)

```
BEGINVTAM
  PORT 23
  SDEFAULTLUS
      SHLU01..SHLU04
  ENDSDEFAULTLUS

  SLUGROUP
      SHRGRP99  SH99LU01..SH99LU04
  ENDSLUGROUP

  LUMAP SHRGRP99 10.1.100.221

  MONITORGROUP SNAONLY
    AVERAGE
    BUCKETS
    NODYNAMICDR
    NOINCLUDEIP
    AVGSAMPPERIOD 120
    AVGSAMPMULTIPLIER 5
    BOUNDARY1 1
    BOUNDARY2 2
    BOUNDARY3 3
    BOUNDARY4 4
  ENDMONITORGROUP

  DESTIPGROUP ALLUSERS    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  MONITORMAP SNAONLY    DESTIPGRP,ALLUSERS

  DEFAULTAPPL SC31TS        ; All users go to TSO
  ALLOWAPPL SC*             ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING SAF TCPIP/SharedRing1       ;keyring shared by servers
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
ENDTELNETPARMS
;
BEGINVTAM
  PORT 992
  DEFAULTLUS
      SC31BS01..SC31BS99
  ENDDEFAULTLUS
```

Figure D-20 (Part 2 of 4) SC31 TN3270B profile (TELNB31C)

```
; ------------------------------------------------------------------
; This NOSSL        group is mapped to use no SSL security.          -
; NOLUSESSIONPEND = Terminate connection upon a logoff
; ------------------------------------------------------------------
  PARMSGROUP NOSSL
   NOLUSESSIONPEND
     CONNTYPE BASIC  ; support non-secure, overrides telnetparms
  ENDPARMSGROUP

; ------------------------------------------------------------------
; The SSLPLAIN     group is mapped to use SSL security               -
;                  with no Client Authentication required           -
; LUSESSIONPEND = Force a requeue back to the USS table upon logoff -
; ------------------------------------------------------------------
  PARMSGROUP SSLPLAIN
     LUSESSIONPEND
     CONNTYPE SECURE ; says plain SSL, no client auth specified
  ENDPARMSGROUP      ; and negotiate all available encryption algorithms

; ------------------------------------------------------------------
; The SSLCERTS     group is mapped to use SSL security               -
;                  and to require Client Authentication (certificates) -
; NOLUSESSIONPEND = Terminate connection upon a logoff from the appl -
; ------------------------------------------------------------------
  PARMSGROUP SSLCERTS
   NOLUSESSIONPEND
     CONNTYPE SECURE       ; Support SSL
     CLIENTAUTH SSLCERT    ; Client Certificate required
     ENCRYPT SSL_DES_SHA   ; use these only, do not consider any others
             SSL_3DES_SHA
     ENDENCRYPT
  ENDPARMSGROUP

; ------------------------------------------------------------------
; The UP2USER      group is mapped to use ANY security (user's choice) -
;                  with no Client Authentication (no certificates)  -
; LUSESSIONPEND = Force a requeue back to the defaultappl upon logoff -
; ------------------------------------------------------------------
  PARMSGROUP UP2USER
     LUSESSIONPEND
     CONNTYPE ANY          ; Whatever User wants to do
  ENDPARMSGROUP

  MONITORGROUP SNAANDIP
    AVERAGE
    BUCKETS
    DYNAMICDR
    INCLUDEIP
    AVGSAMPPERIOD 120
    AVGSAMPMULTIPLIER 5
    BOUNDARY1 1
    BOUNDARY2 2
    BOUNDARY3 3
    BOUNDARY4 4
  ENDMONITORGROUP
```

Figure D-21 (Part 3 of 4) SC31 TN3270B profile (TELNB31C)

```
 DESTIPGROUP GENERALUSER 10.1.8.21     ENDDESTIPGROUP
  DESTIPGROUP ADMIN       10.1.8.22     ENDDESTIPGROUP
  DESTIPGROUP PAYROLL     10.1.8.23     ENDDESTIPGROUP
  DESTIPGROUP SHIPPING    10.1.1.20     ENDDESTIPGROUP
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  MONITORMAP SNAANDIP   DESTIPGRP,GENERALUSER
  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30TS    DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLPO  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER         DESTIPGRP,SHIPPING
  USSTCP USSTABEE,USSSNAEE DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE

  PARMSGROUP DEBUGIT  DEBUG TRACE JOBLOG ENDPARMSGROUP
  IPGROUP DEBUGIPGRP 10.1.100.221
                     10.1.100.222
                     10.1.100.223
                     10.1.100.224
  ENDIPGROUP

  PARMSMAP DEBUGIT  IPGRP,DEBUGIPGRP
 ;---------------------------------------------------------------
 ; There is no DEFAULTAPPL nor USSTCB coded as a default catch all -
 ; So, if any user connects using any other IP address than the    -
 ; four defined by the DESTIPGROUPs above, the Network Solicitor   -
 ; prompt panel will be displayed to that user, in BASIC mode.     -
 ;---------------------------------------------------------------
  ALLOWAPPL SC3*            ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
******************************** Bottom of Data ********************************
```

Figure D-22 (Part 4 of 4) SC31 TN3270B profile (TELNB31C)

SC31 TNLUNS31 primary LUNS PROC for LUNS and LUNR scenario

The PROC JCL for the backup LUNS TNLUNS30 is shown in Figure D-23.

```
 BROWSE    SYS1.PROCLIB(TNLUNS31) - 01.06            Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
//TNLUNS31  PROC PARMS='CTRACE(CTIEZBTN)',
//             PROFILE=LUNS&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//TNLUNS   EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
```

Figure D-23 SC31 TNLUNS31 PROC JCL

We started the TNLUNS31 started task for this scenario by issuing the following command:

```
S TNLUNS31
```

SC31 TNLUNS31 PROFILE for LUNS and LUNR scenario

The profile for the primary LUNS is LUNS31, as shown in Figure D-24.

```
 BROWSE    TCPIPB.TCPPARMS(LUNS31) - 01.41           Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
; ===SC31============ TN3270 Server Profile for LUNS/LUNR =============-
;                                                                        -
;    SSL security.  Sysplex Distribution. LUNS/LUNR.                     -
;                                                                        -
; ---------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
    XCFGROUP
        JOIN
        LUNS 10.1.1.20 4444
        PRIMARY
        RANK 101
    ENDXCFGROUP
    TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
ENDTELNETGLOBALS
****************************** Bottom of Data ******************************
```

Figure D-24 SC31 backup LUNS profile LUNS31

SC31 TCPIPB stack PROC for LUNS and LUNR scenario

Figure D-25 shows that the PROC JCL for the stack is TCPIPB.

```
//TCPIPB    PROC PARMS='CTRACE(CTIEZB00),IDS=00',
//             PROFILE=PROFB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE
//TCPIPB    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM=('&PARMS',
//         'ENVAR("_CEE_ENVFILE=DD:STDENV")')
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(NAMENV31)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure D-25 SC31 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB
```

SC31 TCPIPB stack PROFILE for LUNS and LUNR scenario

Figure D-26 through Figure D-30 on page 455 show that the PROFILE for the stack is PROFB31.

```
 BROWSE    TCPIPB.TCPPARMS(PROFB31) - 01.09          Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
******************************** Top of Data ********************************
NETMONITOR SMFSERVICE
;
; TCPIP.PROFILE.TCPIP
; ===================
;
GLOBALCONFIG
   ECSALIMIT 0M              ; default = 0M
   POOLLIMIT 0M              ; default = 0M
   TCPIPSTATISTICS           ; default = notcpipstatistics
   XCFGRPID 21               ; subplex for TCPIPB on all lpars
   IQDVLANID 21
   SYSPLEXMONITOR DELAYJOIN   RECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   DATAGRAMFWD
   SYSPLEXROUTING
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
   DYNAMICXCF 10.1.7.21    255.255.255.0 8
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
;  RESTRICTLOWPORTS          ; default = unrestrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   NODELAYACKS               ; default = delayacks
;  TTLS                      ; default = NOTTLS
;
UDPCONFIG
;  RESTRICTLOWPORTS          ; default = unrestrictlowports
   UDPCHKSUM                 ; default = udpchksum
   UDPSENDBFRSIZE   65535    ; default = 64K
   UDPRCVBUFRSIZE   65535    ; default = 64K
 NOUDPQUEUELIMIT             ; default = udpqueuelimit
;
SMFCONFIG                    ;
    TYPE118 TCPIPSTATISTICS   ; default = notcpipstatistics
    TYPE119 TCPIPSTATISTICS   ; default = notcpipstatistics
          NOTCPINIT           ; default = notcpinit
          NOTCPTERM           ; default = notcpterm
            FTPCLIENT         ; default = noftpclient
            TN3270CLIENT      ; default = notn3270client
            IFSTATISTICS      ; default = noifstatistics
            PORTSTATISTICS    ; default = noportstatistics
            TCPSTACK          ; default = notcpstack
          NOUDPTERM           ; default = noudpterm
```

Figure D-26 (Part 1 of 5) SC31 TCPIPB profile (PROFB31)

```
; ---------------------------------------------------------------------
 AUTOLOG 5

   OMPB
   FTPDB
   INETDB
;  LPSERVEB
;  RXSERVE
   SENDMAIL
   SNMPDB
   SNMPOSAB
   SNMPQEB
   TRAPFWDB
   RPCBIND JOBNAME RPCBIND1

 ENDAUTOLOG
; ---------------------------------------------------------------------
;INTERFACE OSA20x0I DEFINE IPAQENET (OSA-E) PORTNAME OSA20x0
;TRL MAJ NODE: OSA2080,OSA20A0,OSA20C0,AND OSA20E0
;
INTERFACE OSA2080I
DEFINE IPAQENET
PORTNAME OSA2080
IPADDR 10.1.2.21/24
MTU 1492
VLANID 10
VMAC
;
INTERFACE OSA20A0I
DEFINE IPAQENET
PORTNAME OSA20A0
IPADDR 10.1.2.22/24
MTU 1492
VLANID 10
VMAC
;
INTERFACE OSA20C0I
DEFINE IPAQENET
PORTNAME OSA20C0
IPADDR 10.1.3.21/24
MTU 1492
VLANID 11
VMAC
;
INTERFACE OSA20E0I
DEFINE IPAQENET
PORTNAME OSA20E0
IPADDR 10.1.3.22/24
MTU 1492
VLANID 11
VMAC
;
```

Figure D-27 (Part 2 of 5) SC31 TCPIPB profile (PROFB31)

```
;HiperSockets definitions
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4L    IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5L    IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6L    IPAQIDIO      IUTIQDF6
; Static VIPA definitions                                      -
 DEVICE VIPA1      VIRTUAL 0
 LINK   VIPA1L     VIRTUAL 0   VIPA1
;

 VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set aside some addresses for use with BIND and SIOCSVIPA IOCTL  -
  ;               (10.1.9.21 thru 10.1.9.24)                        -
  ;-------------------------------------------------------------------
   VIPARANGE define move nondisrupt 255.255.255.0  10.1.9.0

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPABACKUP MOVE IMMED 255.255.255.0 10.1.8.25    ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25     PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPABACKUP MOVE IMMED 255.255.255.0 10.1.8.21    ;General
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                         10.1.8.21     PORT 992 20 21 23
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPABACKUP MOVE IMMED 255.255.255.0 10.1.8.22    ;Admin
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.22     PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPABACKUP MOVE IMMED 255.255.255.0 10.1.8.23  ; Payrol
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 992 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
```

Figure D-28 (Part 3 of 5) SC31 TCPIPB profile (PROFB31)

```
;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for  using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPADEFINE 200 MOVE IMMED 255.255.255.0 10.1.8.24  ; EXTRAS
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                         10.1.8.24   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;      Distribute to 10.1.1.10    via IP routing   (   viparoute)  -
  ;      Distribute to 10.1.1.20    via normal XCF   (no viparoute)  -
  ;-------------------------------------------------------------------
   VIPAROUTE      DEFINE 10.1.7.11    10.1.1.10   ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.1.7.21    10.1.1.20   ; sc31's static vipa

 ENDVIPADYNAMIC
;
;
HOME
   10.1.1.20     VIPA1L
   10.1.4.21     IUTIQDF4L
   10.1.5.21     IUTIQDF5L
   10.1.6.21     IUTIQDF6L
;
  PRIMARYINTERFACE VIPA1L
;
PORT
     7 UDP MISCSRVB            ; Miscellaneous Server - echo
     7 TCP MISCSRVB            ; Miscellaneous Server - echo
     9 UDP MISCSRVB            ; Miscellaneous Server - discard
     9 TCP MISCSRVB            ; Miscellaneous Server - discard
    19 UDP MISCSRVB            ; Miscellaneous Server - chargen
    19 TCP MISCSRVB            ; Miscellaneous Server - chargen
    20 TCP OMVS                ; FTP Server
    21 TCP FTPDB1                        ; control port
;   21 TCP FTPDB1  bind 10.1.9.11        ; control port
;   23 TCP OMVS    BIND 10.1.9.23        ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ;   BIND 10.1.8.21
;   23 TCP TNLUNS31 NOAUTOLOG BIND 10.1.1.20
;   25 TCP SMTPB                         ; SMTP Server
;   25 TCP SENDMAIL                      ; Portmap Server
    53 TCP OMVS                          ; Domain Name Server
    53 UDP OMVS                          ; Domain Name Server
   110 TCP INETDB1                        ; Portmap Server
   111 TCP RPCBIND1                      ; Portmap Server
   111 UDP RPCBIND1                      ; Portmap Server
;  123 UDP SNTPD              ; Simple Network Time Protocol Server
;  135 UDP LLBD               ; NCS Location Broker
   161 UDP SNMPDB                        ; SNMP Agent
   162 UDP SNMPQEB                       ; SNMP Query Engine
;  389 TCP LDAPSRV            ; LDAP Server
;  443 TCP HTTPS              ; http protocol over TLS/SSL
;  443 UDP HTTPS              ; http protocol over TLS/SSL
;  512 TCP OMVS BIND 10.1.9.22          ; UNIX REXECD D-VIPA
;  514 TCP OMVS BIND 10.1.9.22          ; UNIX RSHD   D-VIPA
   512 TCP RXSERVEB                     ; TSO  REXECD
```

Figure D-29 (Part 4 of 5) SC31 TCPIPB profile (PROFB31)

```
 514 TCP RXSERVEB                      ; TSO  RSHD
   515 TCP LPSERVEB                      ; LPD Server
   520 UDP OMPB    NOAUTOLOG             ; OMPROUTE
;  580 UDP NCPROUT              ; NCPROUTE Server
   722 TCP CS*                  ; For LPR printing by users CSxx
   723 TCP CS*                  ; For LPR printing by users CSxx
   724 TCP CS*                  ; For LPR printing by users CSxx
   725 TCP CS*                  ; For LPR printing by users CSxx
   726 TCP CS*                  ; For LPR printing by users CSxx
   727 TCP CS*                  ; For LPR printing by users CSxx
   728 TCP CS*                  ; For LPR printing by users CSxx
   729 TCP CS*                  ; For LPR printing by users CSxx
   730 TCP CS*                  ; For LPR printing by users CSxx
   731 TCP CS*                  ; For LPR printing by users CSxx
   750 TCP MVSKERBB                      ; Kerberos
   750 UDP MVSKERBB                      ; Kerberos
   751 TCP ADM@SRVB                      ; Kerberos Admin Server
   751 UDP ADM@SRVB                      ; Kerberos Admin Server
   992 TCP TN3270B NOAUTOLOG   ; Secure TN3270 via internal SSL
  4992 TCP TN3270B NOAUTOLOG   ; Secure TN3270 via AT-TLS
;
; Used for Netview      - needed for AON
SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161

ITRACE OFF

START OSA2080I
START OSA20A0I
START OSA20C0I
START OSA20E0I
START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
;
******************************** Bottom of Data ********************************
```

Figure D-30 (Part 5 of 5) SC31 TCPIPB profile (PROFB31)

SC31 OMPROUTE PROC for LUNS and LUNR scenario

Figure D-31 shows the PROC JCL for OMPROUTE.

```
***************************** Top of Data ***********************
//OMPB   PROC STDENV=OMPENB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=0M,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
***************************** Bottom of Data ********************
```

Figure D-31 SC31 OMPROUTE PROC (OMPB)

SC31 OMPROUTE STDENV file for LUNS and LUNR scenario

Figure D-32 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIP.SC31.STDENV(OMPENB31) - 01.05         Line 00000000 Col 001 080
 Command ===>                                              Scroll ===> CSR
******************************** Top of Data ********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB31)'
OMPROUTE_DEBUG_FILE=/tmp/syslog/debugb31
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure D-32 SC31 OMPROUTE STDENV file (OMPENB31)

SC31 OMPROUTE CONFIG for LUNS and LUNR scenario

The config for OMPROUTE is OMPB31 See Figure D-33 and Figure D-34.

```
BROWSE    TCPIPB.TCPPARMS(OMPB31) - 01.02            Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ********************************
Area Area_Number=0.0.0.2
 Stub_Area=YES
 Authentication_type=None;
;
;New Parameter OSPF
OSPF
 RouterID=10.1.1.20
 Comparison=Type2
 Demand_Circuit=YES;
Global_Options
 Ignore_Undefined_Interfaces=YES
Routesa_Config Enabled=Yes Community="jOs9m2ap" Agent=161;
;
; Static vipa
ospf_interface ip_address=10.1.1.20
           name=VIPA1L
           subnet_mask=255.255.255.0
           Advertise_VIPA_Routes=HOST_ONLY
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=65535;
;
; OSA Qdio 10.1.2.x
ospf_interface ip_address=10.1.2.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=100
           mtu=1492;
; OSA Qdio 10.1.3.x
ospf_interface ip_address=10.1.3.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=90
           mtu=1492;
;
; Hipersockets 10.1.4.x
ospf_interface ip_address=10.1.4.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=1
           attaches_to_area=0.0.0.2
           cost0=80
           mtu=8192;
;
; Hipersockets 10.1.5.x
ospf_interface ip_address=10.1.5.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=1
           attaches_to_area=0.0.0.2
           cost0=80
           mtu=8192;
```

Figure D-33 (Part 1 of 2) SC31 OMPROUTE config (OMPB31)

```
; Dynamic vipa VIPADEFINE
ospf_interface ip_address=10.1.8.*
            subnet_mask=255.255.255.0
            Advertise_VIPA_Routes=HOST_ONLY
            attaches_to_area=0.0.0.2
            cost0=10
            mtu=65535;
;
; Dynamic vipa VIPARANGE
ospf_interface ip_address=10.1.9.*
            subnet_mask=255.255.255.0
            Advertise_VIPA_Routes=HOST_ONLY
            attaches_to_area=0.0.0.2
            cost0=10
            mtu=65535;
;
; Dynamic XCF
interface  ip_address=10.1.7.*
            subnet_mask=255.255.255.0
            mtu=65535;
;
```

Figure D-34 (Part 2 of 2) SC31 OMPROUTE config (OMPB31)

E

FTP and translation tables

This appendix shows you how to work with translation tables in the FTP environment. It covers how to override the default tables for single-byte character sets (SBCS), double-byte character sets (DBCS), and multi-byte character sets (MBCS). Translation support is well documented in the z/OS Communications Server product publication manuals. This appendix does not duplicate that documentation, but instead shows examples of how to use FTP subcommands to manage the translation process.

This section discusses the following translation topics.

Section	Topic
“Conceptual overview of FTP translation” on page 460	General FTP translation support
“Using the RFC2389 and RFC2640 FTP features” on page 465	Special translation features
“Selecting translation tables” on page 467	Examples of using various translation functions

Conceptual overview of FTP translation

z/OS Communications Server is delivered with translation tables supporting many different languages, including both single-byte character sets (SBCS) and double-byte character sets (DBCS). The FTP implementation supports the transfer of Unicode data encoded in UCS-2, UTF-8, UTF-16, and other encoding schemes. Unicode is capable of encoding all of the characters of the major scripts used throughout the world.

What is translation

TCP/IP uses translation tables to convert transmitted data from EBCDIC to ASCII and from ASCII to EBCDIC. TCP/IP provides standard tables that are used as the default if you do not customize your own. When either the z/OS FTP client or the z/OS FTP server is in an FTP session with an ASCII host and you use a data type of ASCII, the data will be translated between EBCDIC and ASCII. This translation takes place on the MVS system.

In addition to single-byte ASCII/EBCDIC conversion, FTP supports DBCS and Unicode conversions to or from EBCDIC.

How does translation work

The FTP.DATA file is read when ftpd is started and the translate tables are set up in the initial address space, which has access to DD statement allocation. A process is then forked to run in the background but an `execv()` is not done at this stage. Therefore, it is an exact copy of the original address space and has all the configuration data. When a client connects, the daemon forks a new address space to handle the client session. In that new address space, an execv() is done to load the ftpdns module, and the configuration parameters are passed in a parameter list over the `execv()`. Figure E-1 shows an overview.

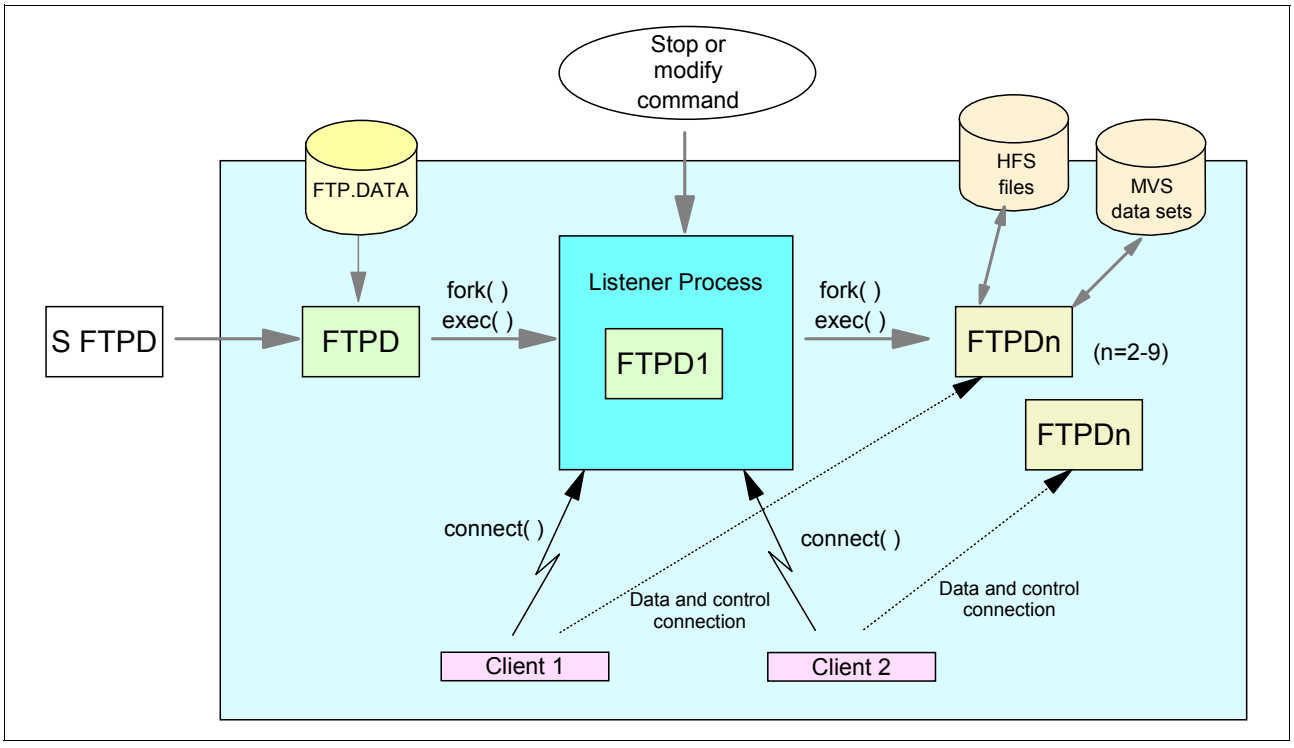


Figure E-1 Process flow of the server

When a new FTP client connects to the FTPD daemon process, the FTP daemon forks another address space that uses the execv() services to start the connection-specific server program, the ftpdns program. When the FTP daemon process forks an FTP server process, a new job name is generated by z/OS UNIX. If the original job name is less than 8 characters, z/OS UNIX adds a digit between 1 and 9. At TCP/IP startup this will always be a 1. So in most cases the FTP daemon address space would be named FTPD1. If your original job name is 8 characters, z/OS UNIX uses the same job name for the background job. Similarly, every instance of the FTP server address space will have a name generated by z/OS UNIX. See *z/OS UNIX System Services Planning*, GA22-7800 for details about how job names are generated in z/OS UNIX.

Because each client gets associated with its own copy of the FTP server, each client can set its own translation parameters and does not adversely affect the other client/server sessions in effect.

SITE and LOCSITE commands and their parameters

Each operating system has unique requirements for allocating files or data sets in its file system. These requirements differ so widely between operating systems that it has been impossible to develop a single protocol that embraces all requirements from all operating systems.

To cover all requirements, the FTP protocol implements a SITE command, which enables an FTP client to send imbedded parameters to the FTP server over the control connection. See Figure E-2.

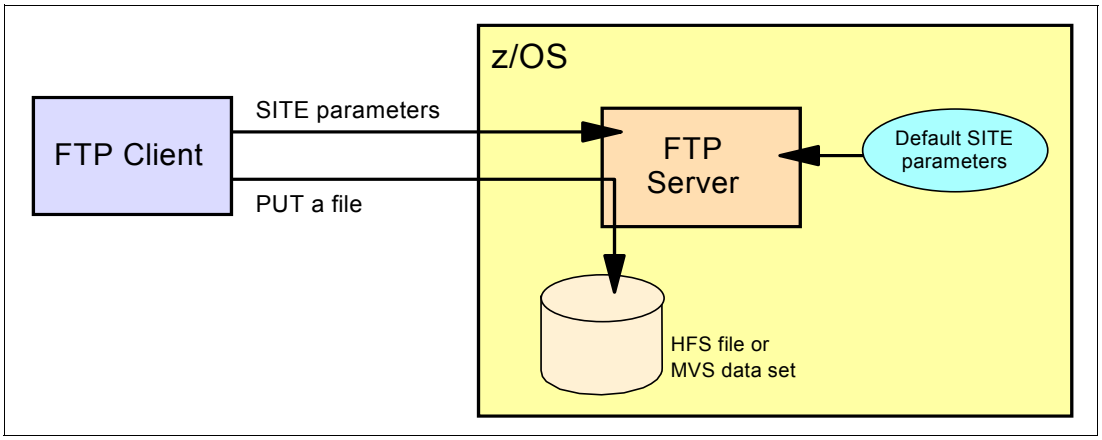


Figure E-2 When to use SITE parameters - z/OS FTP server

When an FTP client issues a `put` to transfer a file to the z/OS FTP server, the FTP server needs specific parameters to allocate a data set. These parameters include record format (RECFM), record length (LRECL), unit-type (UNIT), and blocksize (BLKSIZE), plus many others, depending on the specific operation requested. The FTP server has a set of default values for all the parameters it might need. The client can change many of these values for the current FTP session using the SITE command. See Figure E-3.

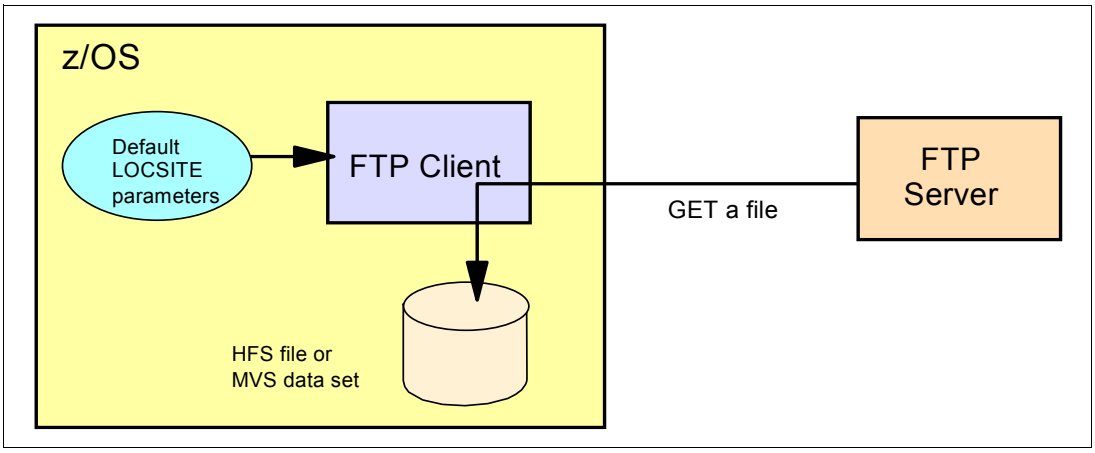


Figure E-3 When to use LOCSITE parameters - z/OS FTP client

If you use the z/OS FTP client function and you retrieve a file from an FTP server somewhere in your IP network, the FTP client function also needs a set of parameters similar to those of the z/OS FTP server, to allocate a data set in MVS. Again a set of default values exists for the z/OS FTP client, but you as a user can change these using the LOCSITE command.

Specification of FTP default values

Default values for data set and disk parameters can be specified using the FTP configuration data sets.

For the FTP client, the default LOCSITE parameters are specified in an installation-wide default configuration data set, or in a user-specific configuration data set. For the FTP server, the default SITE parameters are specified in an FTP server configuration data set.

Default FTP Client LOCSITE parameters

When a user on your z/OS system starts the FTP client function, a set of default local SITE parameters are in effect. The user can change these parameters during the FTP session by using the **`LOCSITE`** command.

The FTP client function searches for a configuration data set that contains the desired default parameters. The search order is defined in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

If none of these data sets is found, the FTP client will use a set of hard-coded default values for the local SITE parameters.

If a user needs to use a different setup, that user can make a temporary modification by means of the LOCSITE command, or a permanent modification by creating a userid.FTP.DATA configuration data set. Review the hlq.SEZAINST(FTCDATA) sample member for client SITE parameters.

Note: Not all SITE parameters can be specified in the FTP.DATA file and not all parameters specified in the FTP.DATA file can be changed with SITE or LOCSITE commands.

For details about both the SITE, LOCSITE, and FTP.DATA client statement parameters, consult *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780.

Default FTP Server SITE parameters (FTPDATA)

The FTP server uses a set of default SITE parameters.

When a client connects to the z/OS FTP server, the default SITE parameters will become the actual SITE parameters for that FTP session, unless the client during the session changes them by means of an FTP client SITE command.

Note: Some systems do not support the SITE command. To pass the SITE parameters to MVS, the user has to enter the QUOTE command in front of SITE. For example, to pass the RECFM=U parameter to MVS, the user might have to type the following command:

```
quote site recfm=u
```

During startup, the server will look for your default SITE parameters in an FTP configuration data set. The search order for FTP.DATA for the server is documented in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

The search step for FTP.DATA that uses a job name as high-level qualifier will use the original job name and not the z/OS UNIX generated FTP daemon job name. This original job name will be used to search for both FTP.DATA and translation tables, for example FTPD.FTP.DATA and FTPD.STANDARD.TCPXLBIN. If the ftpd daemon program were started from the z/OS UNIX shell, these search steps would use the user ID of the process that started the listener program or the value of the _BPX_JOBNAME environment variable. The *datasetprefix* used in the last search step comes from the z/OS UNIX resolver configuration file, in our setup, the SYSTCPD DD statement. If none of these data sets is found, the server will use a set of hard-coded default values.

Review the hlq.SEZAINST(FTPSDATA) sample member for server site parameters. Also review the *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for details about the FTP.DATA configuration statements.

How can FTP translation be applied

TCP/IP uses the following types of translation tables:

- Single byte character set (SBCS) translation tables are used for sing-byte characters. This includes most languages.
- Double-byte character set (DBCS) translation tables are used for converting double-byte characters. DBCS translation tables are required for character sets such as Japanese Kanji, and some Chinese sets, which contain too many characters to represent them by using single-byte codes. SBCS translation tables provide mappings for a maximum of 256 characters. DBCS translation tables can provide up to a theoretical maximum of 65,535 character mappings; however, DBCS character sets usually contain less than this.
- Multi-byte character set (MBCS) translation tables are used in supporting the Chinese standard GB18030. This support is provided by iconv with code page IBM-5488, and does not allow for customized tables. See the ENCODING and MBDATACONN statements in the FTP.DATA file that define this support for FTP in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. Also see *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, to see how to use the LOCSITE and SITE client commands to specify ENCODING and MBDATACONN during an FTP session.

Supported translations

The various languages and encodings supported by FTP (both SBCS and DBCS) are documented in tables in the following product manuals:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780

In a z/OS FTP client session, the user can select the DBCS translation using the DBCS subcommands of the FTP client function. Similarly, the user can issue FTP subcommands to select Unicode data transfer and conversion to/from EBCDIC.

The z/OS FTP server accepts the TYPE subcommand with parameters that can specify DBCS or Unicode translation for the data transfer.

For SBCS translation, the z/OS FTP client user selects a translate table using the FTP command line keyword TRANSLATE, and the z/OS FTP server accepts a SITE command with a translate table name from a remote FTP client.

Translation table search orders

The search order that FTP uses to find a specified translation table are documented in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. A separate search order exists for each of the two data streams: one for SBCS and a another for DBCS.

A user selects DBCS tables independently of SBCS tables. If a remote FTP client wants to transfer a mixed-mode data stream (a data stream that consists of both DBCS and SBCS data sections intermixed with shift-in and shift-out control characters), the user has to select a DBCS translation table using an FTP TYPE command. The user can change an SBCS translation table using a SITE SBDATACONN command. The user can use the selected DBCS table to translate the DBCS sections and the SBCS table to translate the SBCS sections of the mixed-mode data stream.

Both the FTP server and the FTP client allow the use of separate SBCS translation tables for the data connection and the control connection.

If you use the FTP client and want to change the code pages of the client, you have to use the LOCSITE FTP subcommand instead of the SITE subcommand.

Transfer mode and data type

It might seem to be a trivial matter to transfer files between different computer systems, but when you take a closer look, you soon discover a range of issues originating from the diversity of computer architectures represented in a typical IP network. Some operating systems use 7-bit ASCII to represent character data, others use 8-bit ASCII or EBCDIC, just to mention the most obvious. Some operating systems organize files into records; others treat files as continuous streams of data, in some situations without any encoded notion of record boundaries (in such a case, it will be up to the program reading or writing the data to impose a structure onto the data stream).

The FTP protocol deals to a great extent with these issues, but you, as a user of this protocol, have to select the proper options to let FTP transfer a file in such a way that it is usable on the receiving system.

FTP always transfers data in 8-bit bytes; this is called the *transfer size*. If the sending or receiving system uses another byte length, it is up to the FTP client and the FTP server to implement the proper conversion between local byte sizes and the FTP transfer size. When FTP transfers ASCII data, it always transfers it in 8-bit bytes, where the bits are used to encode the ASCII character according to a specific ASCII standard, which is called NVT-ASCII (Network Virtual Terminal ASCII as defined in the TELNET protocol). This implies that when you transfer ASCII type data between two ASCII hosts, a translation from the local ASCII representation to NVT-ASCII for transmission and back to the receiving hosts local ASCII representation always takes place.

When MVS is involved in an ASCII type transfer, MVS will translate the received NVT-ASCII into EBCDIC, and translate data to be transmitted from EBCDIC into NVT-ASCII.

When you request an FTP file transfer, you can characterize the transfer by means of the TYPE attributes. The TYPE attribute is used in the official sources is *data type*, but you might also see the term *transfer type* or *representation type*. Data type is used to signal how the bits of the transmitted data should be interpreted by the receiver.

The data type normally has a value of ASCII, DBCS, and Unicode. We discuss ASCII here.

When you set the data type to ASCII, the receiver knows that the data is character data and that each line of data is terminated using a control sequence of Carriage Return plus Line Feed (CRLF), which in ASCII is X'0D0A'. If MVS is the receiving side, data will be translated from NVT-ASCII to EBCDIC and the CRLF sequences will be removed from the data stream and substituted by traditional MVS record boundaries according to the current settings of the SITE/LOCSITE parameters: RECFM and LRECL. If RECFM is fixed, the data records will be padded with extra spaces to fill up a record. If MVS is the sending side, the data will be translated from EBCDIC into NVT-ASCII and, based on existing record boundaries, CRLF sequences will be constructed and inserted into the ASCII data stream.

A data type of ASCII is the default data type in all FTP implementations.

Using the RFC2389 and RFC2640 FTP features

These RFCs represent the feature negotiation mechanism and FTP internationalization respectively.

RFC 2389: Feature negotiation

The Feature negotiation mechanism enables FTP clients to ask the server which features or options it supports. The following commands are used with this feature:

- On the z/OS Server

  ```
  OPTS
  FEAT
  ```

- On the z/OS Client

  ```
  feature
  ```

- On the non z/OS Client

  ```
  quote feature
  ```

Displaying server features depends upon what is included in the EXTENSIONS statement in FTP.DATA. Figure E-4 shows an example of a FTP.DATA showing some extension statements.

```
EXTENSIONS      MDTM       ; Server can respond to MDTM cmd
EXTENSIONS      UTF8       ; Server can respond to LANG & UTF-8 enc.
EXTENSIONS      REST_STREAM  ; Server can respond to SIZE cmd
```

Figure E-4 FTP.DATA file showing some of the extensions supported

RFC2640: FTP Internationalization

FTP servers use English alpha numerics for their directory and file names, even though the site serves a non-English-speaking community. This is so because FTP constrains pathnames to single byte encodings (ASCII). With Internationalization, 7-bit restrictions on pathnames used in client commands and server responses are removed. UCS transformation format UTF-8 is used, and a new command for language negotiation is defined.

To activate the RFC2640 implementation, the EXTENSIONS UTF8 parameter in TCP is required in FTP.DATA, as shown in Figure E-4 on page 466. This parameter has to be coded in both the z/OS Client and the z/OS Server. A client can thus override statements within FTP.DATA which control code page selection by using the LOCSITE CTRLCONN subcommand.

Requirements to implement these RFCs

RFC 2389 is available in z/OS Communications Server with no additional tasks having to be performed to activate the support.

RFC 2640 requires you to code an EXTENSIONS UTF8 statement in your server and clients FTP.DATA

The National Language Resources component of z/OS Language Environment must be installed for the UTF-8 function to be available.

After UTF-8 has been negotiated between client and server, the STAT command will indicate that UTF-8 is active on the control connection. Figure E-5 shows the results of a successful UTF-8 negotiation.

```
stat
 >>> STAT
 211-Server FTP talking to host 10.12.6.30, port 1088
 211-User: GARTHM  Working directory: GARTHM.
 211-UTF-8 encoding in use on the control connection
 211-The control connection has transferred 309 bytes
 211-There is no current data connection.
 211-The next data connection will be actively opened
 211-to host 10.12.6.30, port 1088,
 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 set to 300
 211-VCOUNT is 59
 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.
```

Figure E-5 STAT command output

Selecting translation tables

If you want to transfer files that contain DBCS data, you have to configure the TCPIP.DATA file with a LOADDBCSTABLES statement with the proper DBCS code names. The following examples show how to select the translation tables. In all of the examples, we configured the following statement in the TCPIP.DATA used by the FTP server and client:

```
DATASETPREFIX TCPIP
LOADDBCSTABLES SJISKANJI EUCKANJI
```

Using the QUOTE SITE subcommand

The FTP server was started by the cataloged procedure named FTPDB, and the FTP daemon address space had a z/OS UNIX-generated name FTPDB1, as shown in Figure E-6.

```
ftp> quote site ctrlconn=7bit   1
200 Site command was accepted
ftp> quote site sbdataconn=garthm.ftpkana.tcpxlbin  2
200 Site command was accepted
ftp> quote type b 1             3
200-Representation type is KANJI Shift-JIS
200 Standard DBCS control used
```

Figure E-6 Selecting translation tables

In this figure, the numbers correspond to the following information:

1. The FTP client changes the FTP server translation table for the control connection to 7-bit ASCII. Because some systems might not recognize the **site ctrlconn** subcommand, you need to use the **quote** FTP subcommand.
2. The FTP client tries to change the translation table for the data connection used by the FTP server to a customized SBCS table.
3. The FTP client asks the FTP server to change the current transfer type to Shift JIS Kanji. The message with the number 200 will be returned to the client, if the FTP server server loads the corresponding DBCS translation table correctly.

If you specify an invalid parameter or the FTP server cannot support the specified translation table, you will see the error message shown in Figure E-7.

```
ftp> quote site ctrlconn=8bit
200-ctrlconn=8bit ignored.  Requested conversion is not supported.
200 Site command was accepted
```

Figure E-7 site ctrlconn command on client

If you specify an invalid table name, the FTP server will return the error message shown in Figure E-8 to the client.

```
ftp> quote site sbdataconn=tcpip.notable
200-Translate file 'tcpip.notable' not found.  SBDATACONN ignored.
200 Site command was accepted
```

Figure E-8 site sbdataconn command on client

If the LOADDBCSTABLES statement is not specified in the TCPIP.DATA file that is allocated for the FTP server, or the proper code pages are not specified to the LOADDBCSTABLES statement, the attempt will fail and the client will receive the message shown in Figure E-9.

```
504-Type not Supported. Translation table not Loaded.
504 Type remains Ascii NonPrint
```

Figure E-9 Type not supported error

Using the TRACE option at the server

If you set the TRACE option for the z/OS FTP server, the trace entries shown in Figure E-10 are written to the `syslog` file (if daemon.debug has been defined in `/etc/syslog.conf`) or displayed on the MVS console if syslogd is not active.

```
get_command: select rc is 1
get_command: received 20 bytes
Parse_cmd: command line: site ctrlconn=7bit.  1
site_cmd entered with 'ctrlconn=7bit'
site(): processing ctrlconn=7bit
site: freeing site arg: 'CTRLCONN'
get_command: select rc is 1
get_command: received 40 bytes
Parse_cmd: command line: site sbdataconn=kakky.ftpkana.tcpxlbin.  2
site_cmd entered with 'sbdataconn=kakky.ftpkana.tcpxlbin'
site(): processing sbdataconn=kakky.ftpkana.tcpxlbin
site: freeing site arg: 'SBDATACONN'
get_command: select rc is 1
get_command: received 10 bytes
Parse_cmd: command line: type b 1.
xtype: xtype routine entered with type 'B' dataformat '1' opt1 ' ' opt2
trying to open DBCS file //'FTPDB.SRVRFTP.TCPKJBIN'      3
fopen failed for //'FTPDB.SRVRFTP.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
trying to open DBCS file //'TCP.SRVRFTP.TCPKJBIN'        3
fopen failed for //'TCP.SRVRFTP.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
trying to open DBCS file //'FTPDB.STANDARD.TCPKJBIN'     3
fopen failed for //'FTPDB.STANDARD.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
NC1444 trying to open DBCS file //'TCP.STANDARD.TCPKJBIN' 3
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
EZY2720I Using Japanese translation tables in 'TCP.STANDARD.TCPKJBIN' 4
```

Figure E-10 syslogd output

In this figure, the numbers correspond to the following information:

1. The FTP client changes the FTP server translation table for the control connection to 7-bit ASCII.
2. The FTP client changes the FTP server translation table for the single byte data connection to a private translate table.
3. You can see the search order for the DBCS translation table. In this case, the name of the FTPD cataloged procedure is FTPDB, so FTPDB will be used as the original job name.
4. The DBCS translation table loaded by the FTP server.

Using the DEBUG option at the client

Figure E-11 shows examples of messages displayed when we started FTP clients with the debug option. You can use either **-d** or (TRACE for this option).

```
EZA1736I FTP -d (EXIT
EZYFT18I Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages. 1
EZY2640I Using dd:SYSFTPD for local site configuration parameters.
EZA2794I Both CCTRANS and CTRLCONN were specified. Using CTRLCONN.        2
                                        CCTRANS will be ignored.
EZA2795I Both SBTRANS and SBDATACONN were specified. Using SBDATACONN.    3
                                        SBTRANS will be ignored.
EZYFT27I Using conversion between 'IBM-850' and 'IBM-1047'
                                        for the control connection.      4
EZYFT31I Using TCP.FTPKANA.TCPXLBIN for FTP translation tables
                                        for the data connection.         5
EP4873 set_dbcs: __ipdbcs() returned 2 parms from LOADDBCSTABLES statement(s)
EP4879 main: dbcstables->__ip_dbcs_listÿ0À is: SJISKANJ.  Len is: 8       6
EP4879 main: dbcstables->__ip_dbcs_listÿ1À is: EUCKANJI.  Len is: 8       6
EZA1450I IBM FTP CS/390 296 00:10 UTC
EZA1466I FTP: using TCPIPA instead of INET
EZA1772I FTP: EXIT has been set.
EZA1456I Connect to ?
EZA1736I 10.24.104.26
```

Figure E-11 FTP client with debug option

In this figure, the numbers correspond to the following information:

1. The messages used by the FTP clients are stored in message catalogs to allow message translation.
2. If both the CCTRANS and CTRLCONN keywords are specified in FTP.DATA, CTRLCONN is preferred.
3. If both the SBTRANS and SBDATACONN keywords are specified in FTP.DATA, SBDATACONN is preferred.
4. The translation used by the client for control connection.
5. The translation table used by the client for data connection.
6. The DBCS code pages specified in the TCPIP.DATA configuration data set have been allocated for this client. Each code page name is recognized by the preceding 8 characters.

Using the TRANSLATE sub command

If you specify the TRANSLATE FTP command option, the *hlq*.STANDARD.TCPXLBIN data set is never used. If the translation table search procedure using a data set name specified with this option fails, then FTP ends with an error message as shown in Figure E-12.

```
KAKKY@/u/kakky$ftp -d 10.24.104.47 \(translate notable  1
Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
Using FTP configuration defaults.
NXO57O read_xlate_files: Unable to open /u/kakky/notable.tcpxlbin :  2
                 EDC5129I No such file or directory.
NXO613 read_xlate_files: Unable to open //'KAKKY.NOTABLE.TCPXLBIN' :
                 EDC5049I The specified file name could not be located.
NXO625 read_xlate_files: Unable to open //'TCP.NOTABLE.TCPXLBIN' :
                 EDC5049I The specified file name could not be located.
Cannot load translate table specified by TRANSLATE parameter notable 3
```

Figure E-12 Translate table not found

In this figure, the numbers correspond to the following information:

1. If you issue the FTP command in the z/OS UNIX shell with MVS style options, you have to precede the left parenthesis with an escape character such as the backslash(\).
2. In the z/OS UNIX environment, the translation table in the Hierarchical File System is the top of the table search hierarchy.
3. The FTP command stops. No more table search attempts are executed.

Setting a DBCS transfer mode

When you use the DBCS data transfer mode in the FTP client, the debug option of the FTP client shows you the messages displayed in Figure E-13.

```
EZA1460I Command:
sjiskanji (notype  1
 PC0291 Input to parCmd is :
 PC0293 parseCmd: Number of parameters is 2
 PC0296 parseCmd: parameter 0 is sjiskanji
 PC0296 parseCmd: parameter 1 is (notype
 PC0486 fndCmd: entering with sjiskanji.
 PC0568 fndCmd: Command found is sjiskanji
 PC0305 parseCmd: fndCmd returned the cmdrecord for sjiskanji
 PC0386 parseCmd: Using primary session
 CT2205 sjiskj: routine entered with parmcount=2
 NC1017 get_client_dbcs_table() entered for lang type 1
 NC1053 trying to open DBCS file //'KAKKY.FTP.TCPKJBIN'          2
 NC1079 fopen failed for //'KAKKY.FTP.TCPKJBIN'.
               EDC5049I The specified name could not be located.
 NC1053 trying to open DBCS file //'TCP.FTP.TCPKJBIN'            2
 NC1079 fopen failed for //'TCP.FTP.TCPKJBIN'.
               EDC5049I The specified file could not be located.
 NC1053 trying to open DBCS file //'KAKKY.STANDARD.TCPKJBIN'     2
 NC1079 fopen failed for //'KAKKY.STANDARD.TCPKJBIN'.
               EDC5049I The specified file name could not be located.
 NC1053 trying to open DBCS file //'TCP.STANDARD.TCPKJBIN'       2
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1061 Using Japanese translation tables in 'TCP.STANDARD.TCPKJBIN' 3
 CT0334 entering cliDBOptopns() for newftpoptformat=0, parmcount=2
 CT0711 cliDBOpt: no cmd sent to server
 PC0452 parseCmd: Using primary session.
 EZA1460I Command:
```

Figure E-13 Debug log when using DBCS transfer mode

In this figure, the numbers correspond to the following information:

1. Change the data transfer mode to Japanese Shift JIS Kanji. To suppress the sending of the corresponding TYPE message to the FTP server which might not recognize it, the user issues the SJISKANJI FTP subcommand with the NOTYPE parameter.
2. The debug message shows you the DBCS translation search order used by the FTP client.
3. The DBCS translation table loaded by the FTP client.

Enabling Unicode transfer mode

Figure E-14 illustrates how to enable Unicode data transfer mode. In this example, we used a PC FTP client and z/OS FTP server.

```
ftp> quote type u 2   1
200 Representation type is UCS-2
ftp> quote site noucstrunc noucssub ucshostcs=ibm-939  2
200 Site command was accepted
ftp> put jpdata.ucs 'kakky.test.jpdata'
200 Port request OK.
125 Storing data set KAKKY.TEST.JPDATA
250 Transfer completed successfully.
local: jpdata.ucs remote: 'kakky.test.jpdata'
96 bytes sent in 0.00 seconds (0 Kbytes/s)
```

Figure E-14 Unicode data transfer mode

In this figure, the numbers correspond to the following information:

1. Issue TYPE U 2. Receiving this TYPE command, the FTP server sets the transfer type to UCS-2 TYPE. Because the PC client cannot recognize this TYPE, you have to precede the FTP subcommand with QUOTE.
2. Specify the EBCDIC code set to be used for converting to/from Unicode. In this sample, we use the IBM-939 code character set. You have to specify the EBCDIC code set using the code character set name supported by the iconv() API. At the same time, we disable the substitution and truncation. If you transfer the EBCDIC data that contains DBCS characters, you should disable the truncation, or the EBCDIC data might collapse.

Verify the server status using STAT FTP subcommand. The output shown in Figure E-15 is part of the output from the STAT subcommand.

```
211-using Mode Stream, Structure File, type UCS-2, byte-size 8
211-TYPE U data will be converted to/from IBM-939       1
211-UCS Substitution: OFF, UCS Truncation: OFF          2
211-Byte Order: big-endian                              3
```

Figure E-15 Partial STAT command output

In this figure, the numbers correspond to the following information:

1. The data encoded in Unicode will be converted to the IBM-939 character set.
2. Neither the UCS substitution nor UCS truncation are enabled. If you transfer SBCS/DBCS mixed data, then disable the UCS truncation.
3. The byte order for the Unicode encoding is set to big-endian by default. You can change it to the little-endian byte order by issuing the FTP subcommand: TYPE U 2 L.

If you want to use the z/OS FTP client in Unicode transfer type, you have to issue the **UCS2** FTP subcommand and optionally the LOCSITE subcommand with UCS* parameters before starting the file transfer. You can also verify the status of the FTP client with the LOCSTAT FTP subcommand.

F

Our implementation environment

We wrote the four *z/OS Communications Server TCP/IP Implementation* books at the same time. Given the complexity of this project, we needed to be creative in organizing the test environment so that each team could work with minimal coordination and interference from the other teams. In this appendix, we show the complete environment that we used for the four books, and the environment that we used for this book.

The environment used for all four books

To enable concurrent work on each of the four books, we set up and shared the test environment illustrated in Figure F-1.

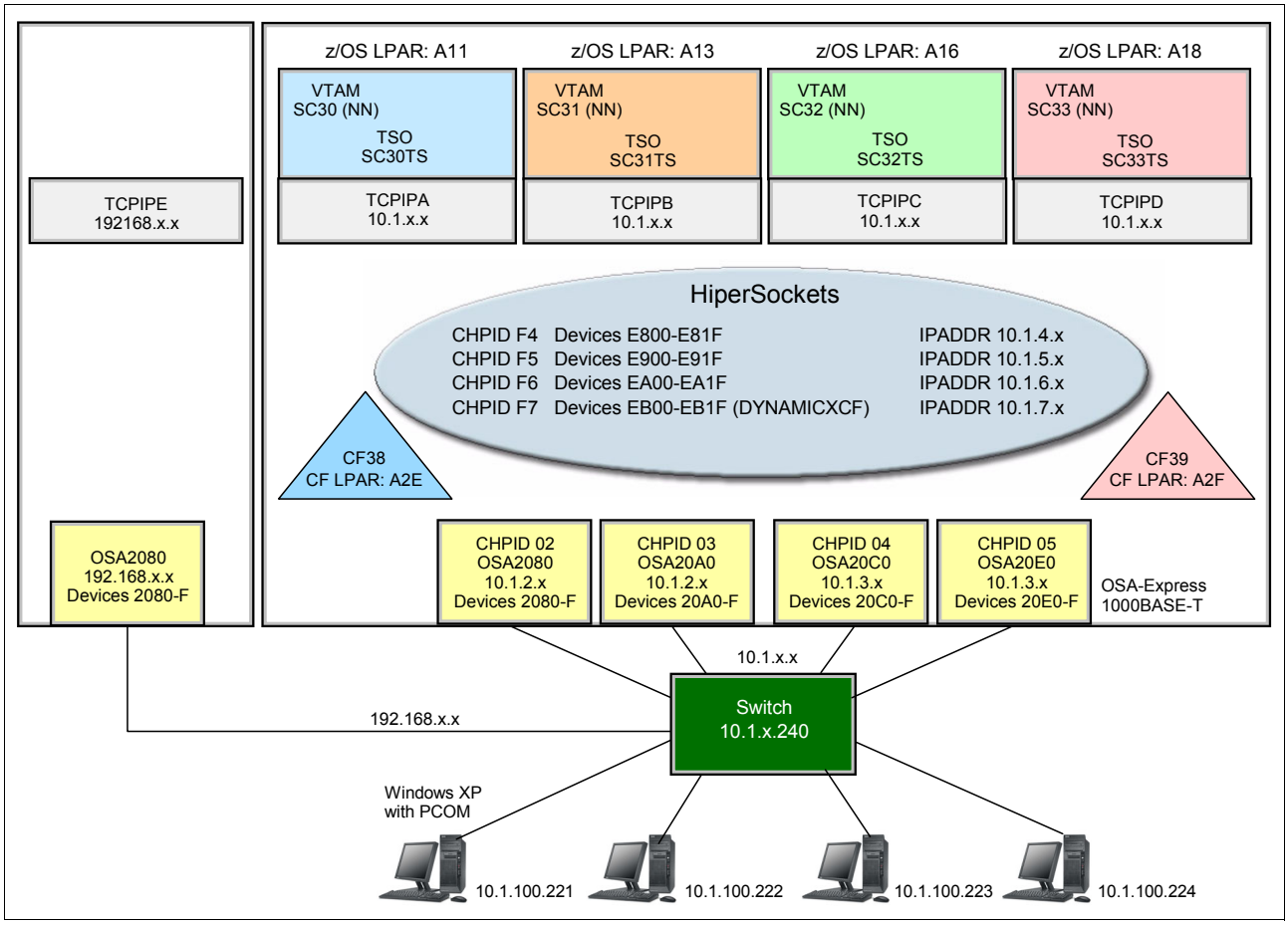


Figure F-1 Our implementation environment

We wrote our books (and ran our implementation scenarios) using four logical partitions (LPARs) on an IBM System z196-32 (referred to as LPARs A11, A13, A16, and A18). We implemented and started one TCP/IP stack on each LPAR. Each LPAR shared the following resources:

- HiperSockets inter-server connectivity
- Coupling facility connectivity (CF38 and CF39) for Parallel Sysplex scenarios
- Eight OSA-Express3 1000BASE-T Ethernet ports connected to a switch

Finally, we shared four Windows workstations, representing corporate network access to the z/OS networking environment. The workstations are connected to the switch. For verifying our scenarios, we used applications such as TN3270 and FTP.

The IP addressing scheme that we used allowed us to build multiple subnetworks so that we would not impede ongoing activities from other team members.

VLANs were also defined to isolate the TCP/IP stacks and portions of the LAN environment (Figure F-2).

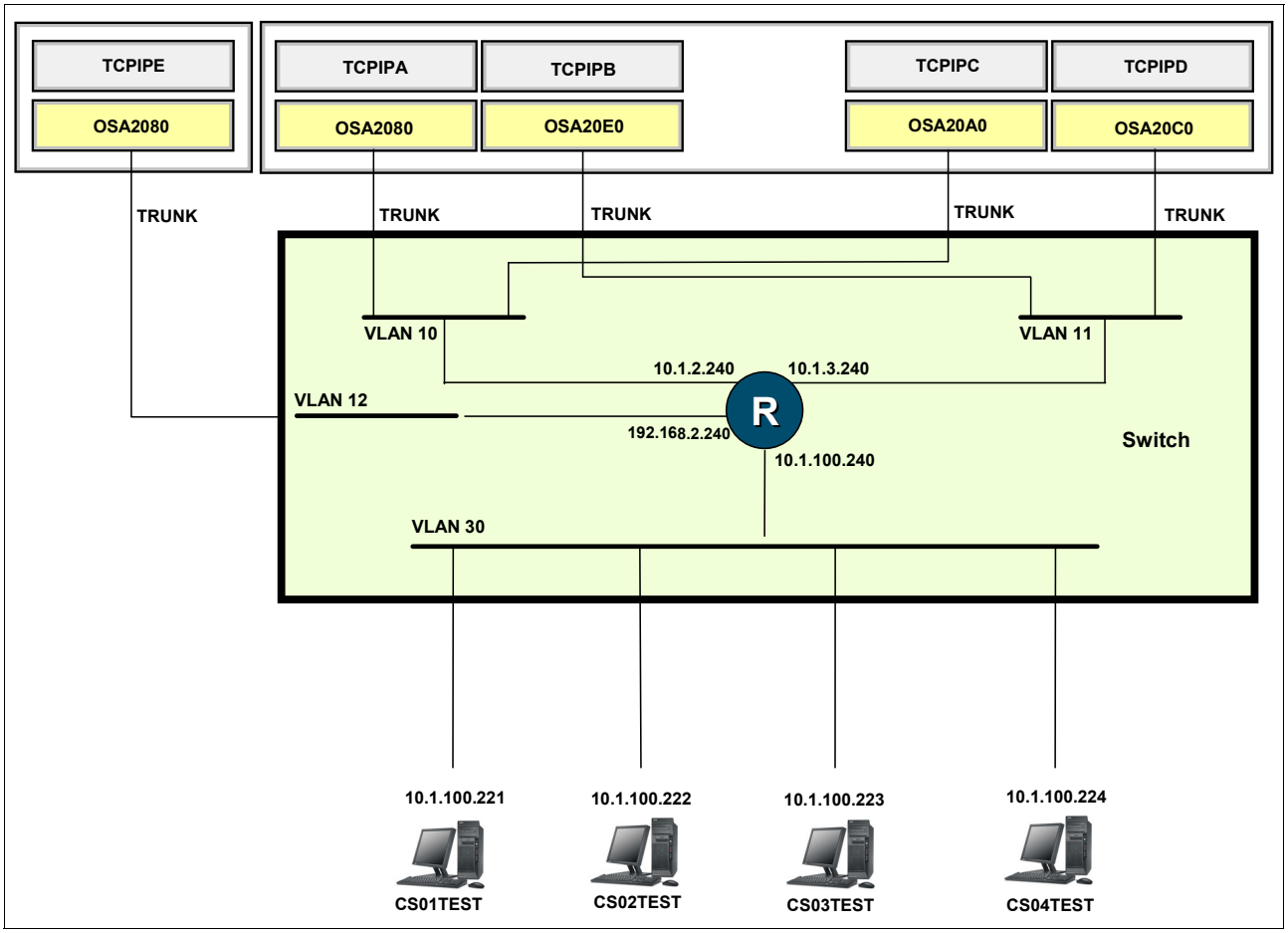


Figure F-2 LAN configuration - VLAN and IP addressing

Our focus for this book

Figure F-3 depicts the environment that we worked with, as required for our basic function implementation scenarios.

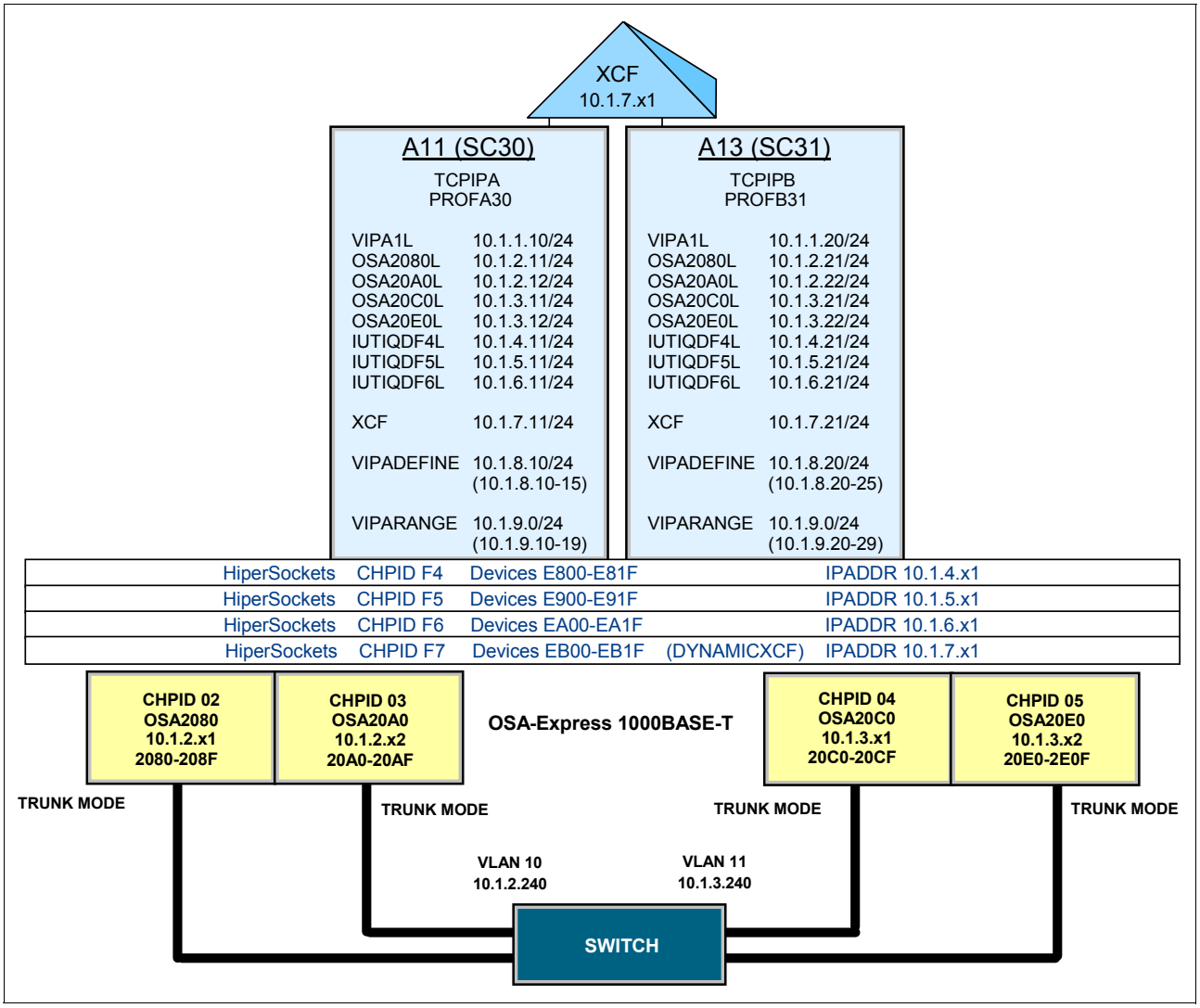


Figure F-3 Our environment for this book

Related publications

These publications are particularly suitable for a more detailed discussion of the topics that are covered in this book.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 481. Note that some of the documents referenced here might be available in softcopy only.

- *High Availability Considerations: SAP R/3 on DB2 for OS*, SG24-2003
- *HiperSockets Implementation Guide*, SG24-6816
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999
- *IP Network Design Guide*, SG24-2580
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *OSA-Express Implementation Guide*, SG24-5948
- *SNA in a Parallel Sysplex Environment*, SG24-2113
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *z/OS 1.6 Security Services Update*, SG24-6448

Other publications

The following publications are also relevant as further information sources:

- *z/OS XL C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS MVS IPCS Commands*, SA22-7594
- *z/OS MVS System Commands*, SA22-7627
- *z/OS TSO/E Command Reference*, SA22-7782
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS Communications Server: CSM Guide*, SC31-8808
- *z/OS Communications Server: New Function Summary*, GC31-8771
- *z/OS Communications Server: Quick Reference*, SX75-0124
- *z/OS Communications Server: IP and SNA Codes*, SC31-8791
- *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference*, SC31-8788
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IPv6 Network and Application Design Guide*, SC31-8885
- *z/OS Migration*, GA22-7499
- *z/OS MVS System Commands*, SA22-7627
- *OSA-Express Customer's Guide and Reference*, SA22-7935
- *z/OS Communications Server: SNA Operation*, SC31-8779
- *z/OS TSO/E Command Reference*, SA22-7782
- *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services File System Interface Reference*, SA22-7808
- *z/OS UNIX System Services Messages and Codes*, SA22-7807
- *z/OS UNIX System Services Parallel Environment: Operation and Use*, SA22-7810
- *z/OS UNIX System Services Programming Tools*, SA22-7805
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS UNIX System Services User's Guide*, SA22-7801
- *sendmail, 4th Edition* by Costales, Assmann, Jansen, Shapiro, from O'Reilly Media, Inc. (ISBN 10: 0-596-51029-2)
- *termcap & terminfo*, by Mui, O'Reilly, Stran, from O'Reilly Media, Inc. (ISBN 10: 0-937175-22-6)

Online resources

The following websites are also relevant as further information sources:

- Mainframe networking

  http://www.ibm.com/servers/eserver/zseries/networking/

- z/OS Communications Server support

  http://www-306.ibm.com/software/network/commserver/zos/support/

- z/OS Communications Server product overview

  http://www.ibm.com/software/network/commserver/zos/

- z/OS Communications Server publications

  http://www-03.ibm.com/systems/z/os/zos/bkserv/r9pdf/#commserv

How to get IBM Redbooks publications

You can search for, view, or download Redbooks publications, Redpapers publications, Hints and Tips, draft publications, and Additional materials, as well as order hardcopy Redbooks publications or CD-ROMs, at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Redbooks

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 2 Standard Applications

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 2 Standard Applications

Redbooks®

Provides information about z/OS Communications Server TCP/IP standard applications

Discusses how to take advantage of TCP/IP standard applications for your needs

Includes TCP/IP application implementation examples

For more than 40 years, IBM mainframes have supported an extraordinary portion of the world's computing work, providing centralized corporate databases and mission-critical enterprise-wide applications. The IBM System z, the latest generation of the IBM distinguished family of mainframe systems, has come a long way from its IBM System/360 heritage. Likewise, its IBM z/OS operating system is far superior to its predecessors, providing, among many other capabilities, world-class, state-of-the-art, support for the TCP/IP Internet protocol suite.

TCP/IP is a large and evolving collection of communication protocols managed by the Internet Engineering Task Force (IETF), an open, volunteer, organization. Because of its openness, the TCP/IP protocol suite has become the foundation for the set of technologies that form the basis of the Internet. The convergence of IBM mainframe capabilities with Internet technology, connectivity, and standards (particularly TCP/IP) is dramatically changing the face of information technology and driving requirements for ever more secure, scalable, and highly available mainframe TCP/IP implementations.

The IBM z/OS Communications Server TCP/IP Implementation series provides understandable, step-by-step guidance about how to enable the most commonly used and important functions of z/OS Communications Server TCP/IP.

This IBM Redbooks publication provides useful implementation scenarios and configuration recommendations for many of the TCP/IP standard applications that z/OS Communications Server supports.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7997-00 0738436313