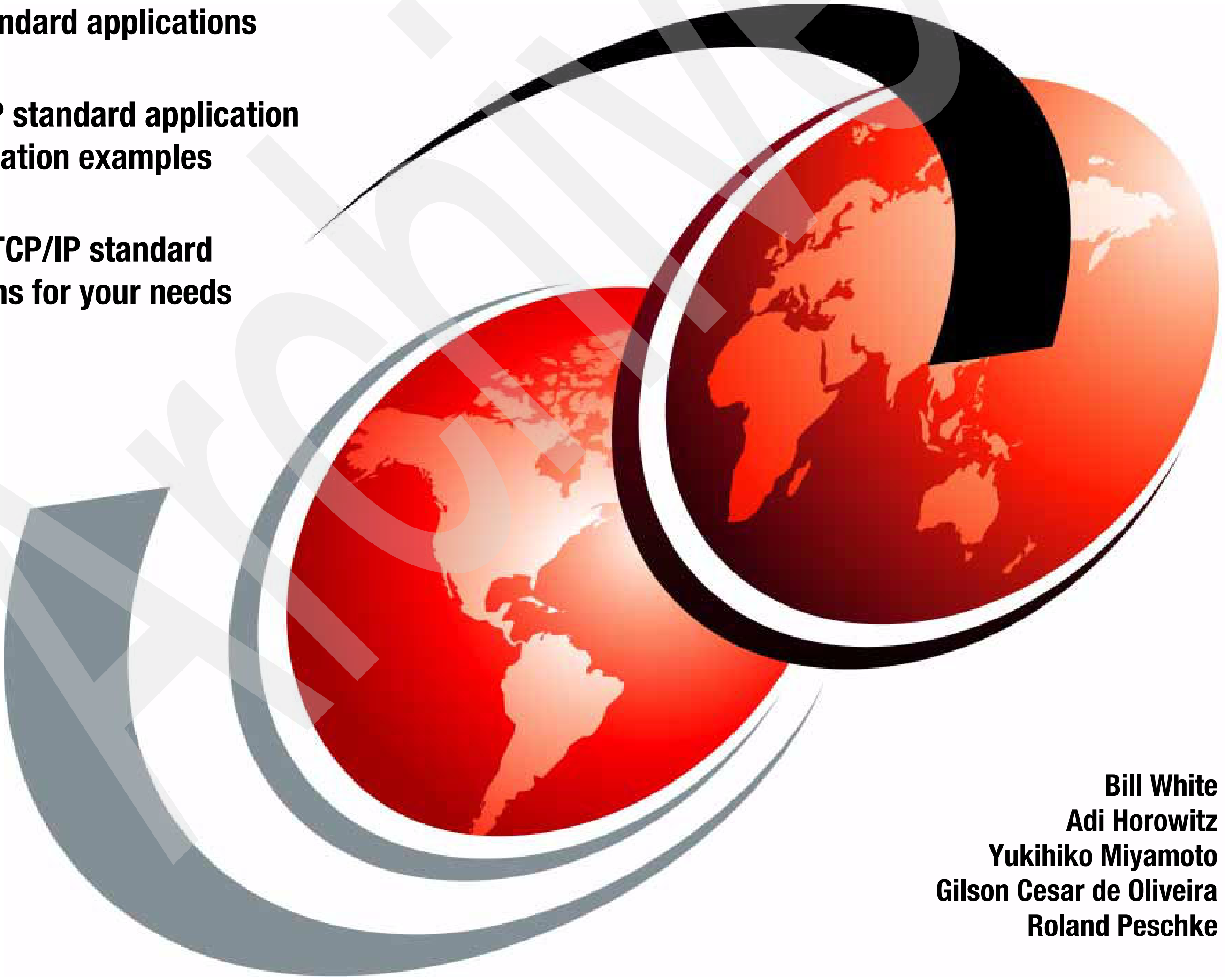
Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications

Understand important CS for z/OS TCP/IP standard applications

See TCP/IP standard application implementation examples

Leverage TCP/IP standard applications for your needs



Bill White
Adi Horowitz
Yukihiko Miyamoto
Gilson Cesar de Oliveira
Roland Peschke

Redbooks

ibm.com/redbooks

International Technical Support Organization

Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications

December 2006

SG24-7340-00

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

First Edition (December 2006)

This edition applies to Version 1, Release 8 of Communications Server for z/OS.

Contents

Notices

Trademarks

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

AFP™
AIX®
C/370™
CICS®
Domino®
DPI®
eServer™
HiperSockets™
Infoprint®
IBM®
ibm.com®
IMS™
IP PrintWay™
Language Environment®
Lotus Notes®
Lotus®
MVS™
NetSpool™
NetView®
Notes®
Print Services Facility™
PrintWay™
Redbooks™
Redbooks (logo) ™
Resource Link™
RACF®
S/370™
System z™
System z9™
System/360™
Tivoli®
VTAM®
WebSphere®
z/OS®
zSeries®
z9™

The following terms are trademarks of other companies:

Java, SNM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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 z9™, the latest generation of the IBM distinguished family of mainframe systems, has come a long way from its IBM System/360™ heritage. Likewise, its 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 that are 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.

This new and improved IBM Redbook series *Communications Server (CS) for z/OS TCP/IP Implementation* provides easy to understand, step-by-step guidance on enabling the most commonly used and important functions of CS for z/OS TCP/IP.

In *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications*, SG24-7340, we provide useful implementation scenarios and configuration recommendations for many of the TCP/IP standard applications supported by the z/OS Communications Server.

For more specific information about CS for z/OS base functions, high availability, and security, see the other volumes in the series. These are:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342

Our implementation environment

We wrote the four books in the *Communications Server for z/OS V1R8 Implementation* guides at the same time. Given the complexity of our test environment, we needed to be somewhat creative in organizing the environment so that each team could work with minimal coordination and interference from the other teams.

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 1 on page xii.

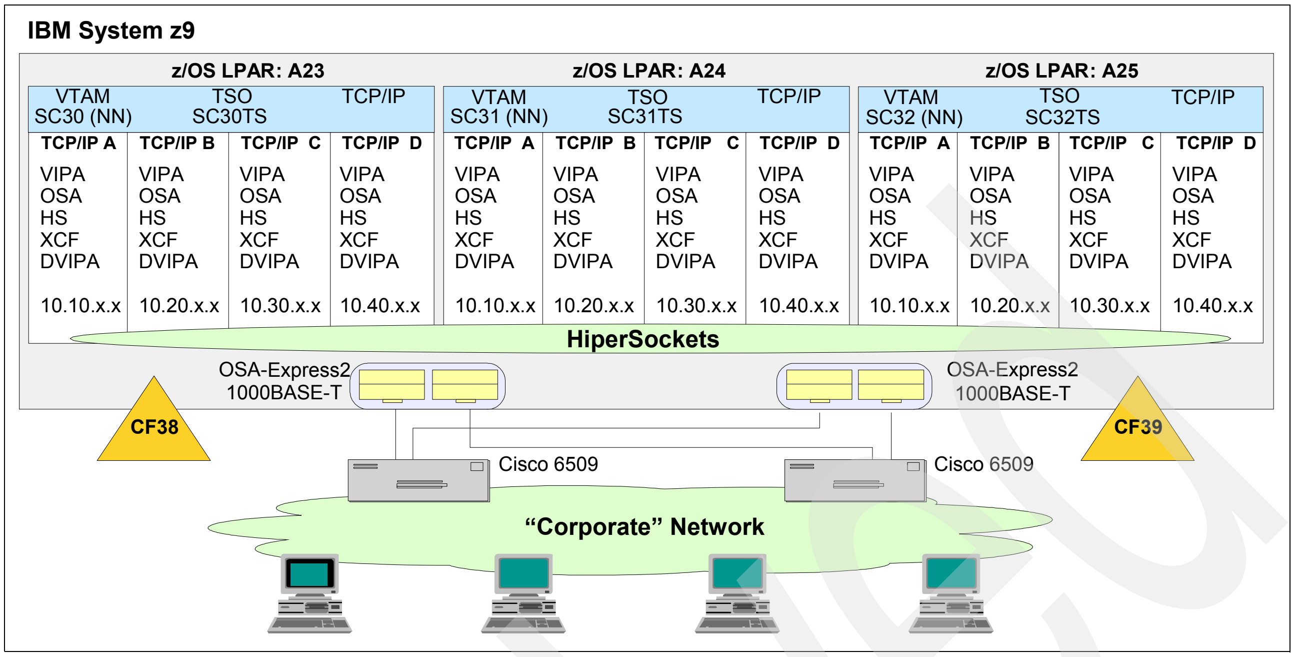


Figure 1 Our implementation environment

Our books were updated (and our implementation scenarios were run) using three logical partitions (LPARs) on an IBM System z9-109 (referred to as A23, A24, and A25). Because we worked on four books at the same time, we implemented *four* TCP/IP stacks on each LPAR (admittedly, a configuration not recommended for a production environment, but convenient for our purposes). Each LPAR shared:

- HiperSockets™ connectivity
- Coupling Facility connectivity (CF38 and CF39) for parallel sysplex scenarios
- Four OSA-Express2 1000BASE-T Ethernet ports cross-connected to a pair of Cisco 6509 switches

Finally, we shared 10 workstations, representing corporate network access to the z/OS networking environment, for scenario verification (using applications such as TN3270 and FTP).

Our IP addressing convention is as follows:

- The first octet is the network (always 10 for our environment).
- The second octet is the VLAN (10, 20, 30, 40) assigned to the stack. (Essentially, except when required by a specific implementation scenario, each team's stacks shared a common VLAN.)
- The third octet refers to the device:
  - The addresses with the third octet of 2 or 3 are defined to the OSA devices.
  - The addresses with the third octet of 4, 5, or 6 are defined to the HiperSocket devices.

Important: Our use of multiple TCP/IP stacks on each LPAR and our TCP/IP addressing were set up for our convenience and are not recommended approaches for your environment.

Our focus for this book

Given the above actual environment, the (simplified) environment that we had to work with for this book is illustrated in Figure 2.

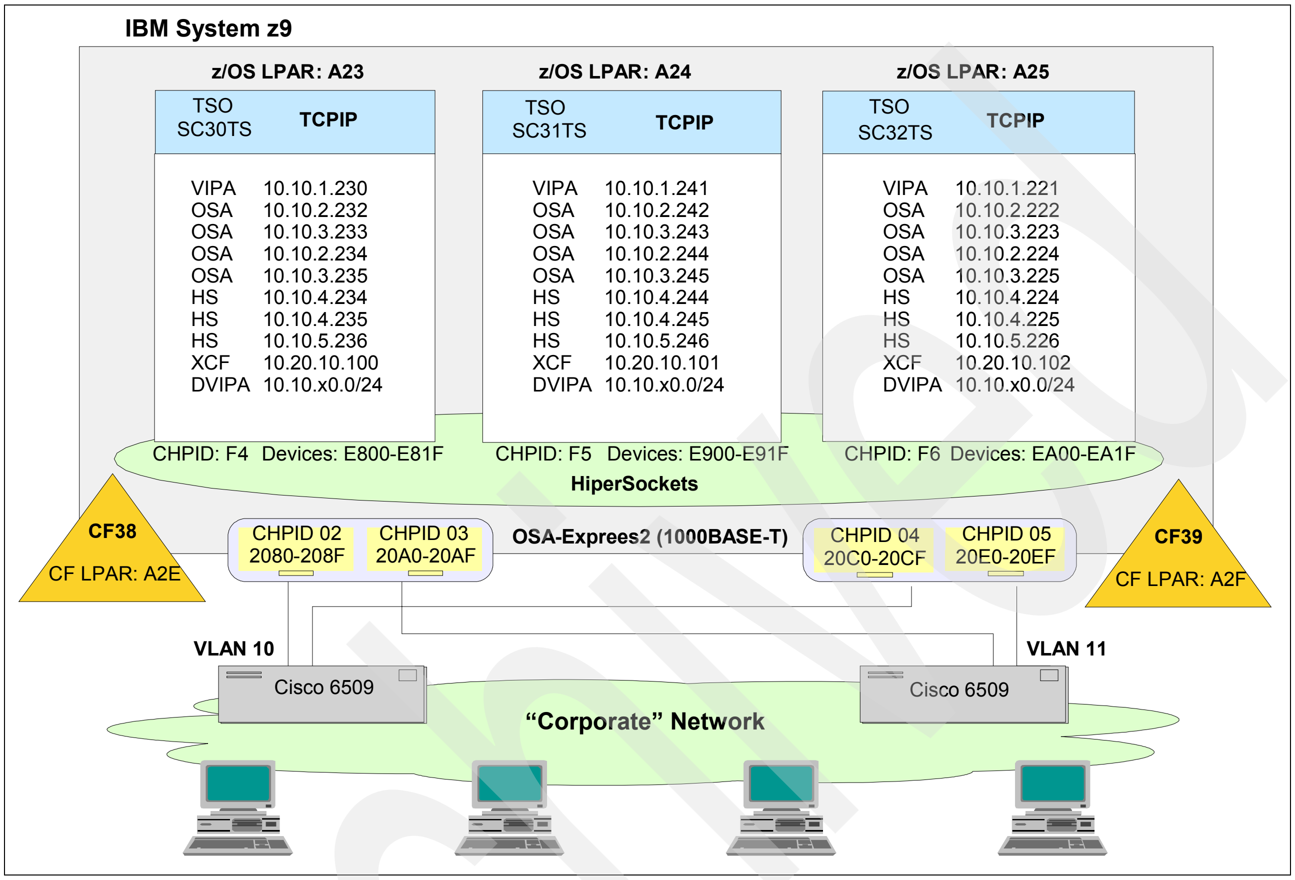


Figure 2 Our environment for this book

For the purposes of this book we viewed the environment as three LPARs leveraging coupling facilities, HiperSockets, and OSA connectivity as required for our implementation scenarios.

The team that wrote this redbook

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

Bill White is a Project Leader and Senior Networking Specialist at the International Technical Support Organization, Poughkeepsie Center.

Adi Horowitz is an IT Specialist who has worked for the past eight years in enterprises that require intensive computing resources, such as the Israel Defence Forces Data Center and Tel-Aviv University. Her areas of expertise include z/OS, Linux® on System z™, and IBM Communications Server for z/OS. She is well-versed with other operating systems and products, specializing in communication products and information systems.

Yukihiko Miyamoto is an IT Specialist in IBM Japan. For ten years, he has specialized in WAN, LAN, TCP/IP, and SNA, with a primary focus on router and switch networking technologies. He currently works with z/OS Communications Server as a Technical Support member. He holds a master's degree in Systems Engineering from Hiroshima University.

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 15 years of experience in mainframe networking with expertise in VTAM®,NCP, TCP/IP, CS for z/OS, IBM 3745/46 and OSA-Express.

Roland Peschke is a Senior IT Networking Specialist providing z/OS Communications Server (TCP/IP and SNA/APPN) consulting and education services to IBM customers. His comprehensive experiences in these areas are based on more than three decades of employment with IBM Germany and ITSO Raleigh. He worked intensively on several SNA and TCP/IP Redbooks, which include load balancing solutions in z/OS Sysplex environments.

Thanks to the following people from the International Technical Support Organization, Poughkeepsie Center for their contributions to this project: David Bennin, Rich Conway, Mike Connolly, and most importantly, Bob Haimowitz, Greg Geiselhart, and Bill Ogden.

As is always the case with any complex technical effort, success would not have been possible without the advice, support, and review of many outstanding technical professionals. We are especially grateful for the significant expertise and contributions of content to this book from the Communications Server for z/OS Development team, especially from Kristine Adamson, Alfred Christensen, Mike Fox, Gary McAfee, and Doug Trottman.

Finally, we want to thank the authors of the previous *Communications Server (CS) for z/OS TCP/IP Implementation* series of Redbooks for creating the groundwork for this series of CS Redbooks; the authors include: Rama Ayyar, Valirio Braga, Octavio Ferreira, Michael Jensen, Sherwin Lake, Bob Louden, Garth Madella, Joel Porterie, Marc Price, Larry Templeton, and Thomas Wienert.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook 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

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 V1R8 Communications Server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains comprehensive descriptions of the individual parameters for setting up a TN3270E Telnet server. It also includes step-by-step checklists and supporting examples. It is not the intent of this book to duplicate the information in the above-referenced manual, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 1.1.2, “Additional information sources” on page 3.

This chapter discusses the following.

Section	Topic
1.1, “Overview” on page 2	Discusses the basic concepts of TN3270
1.2, “Why TN3270 is important” on page 4	Discusses key characteristics of TN3270 and why it may be important in your environment
1.3, “The common design scenarios for TN3270” on page 7	Presents commonly implemented TN3270 design scenarios, their dependencies, advantages, considerations, and our recommendations
1.4, “How a TN3270E server is implemented” on page 25	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

1.1 Overview

As illustrated in Figure 1-1, TN3270E Telnet server is one of the standard applications provided with the z/OS Communications Server.

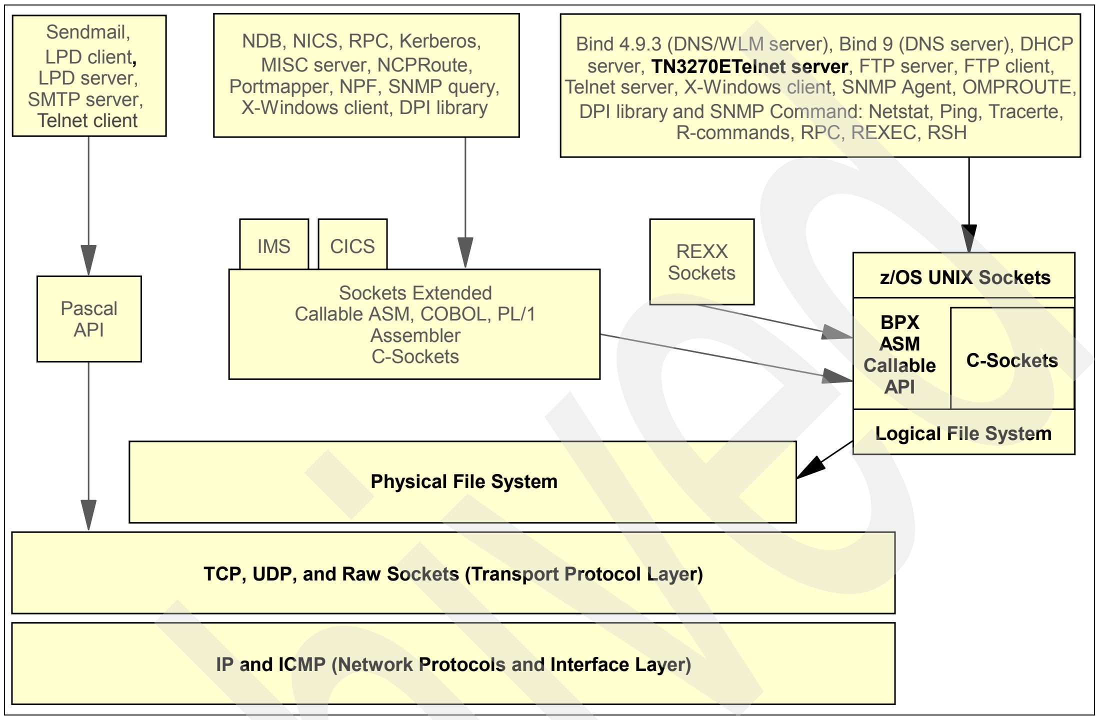

Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270ETelnet server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
REXX Sockets
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
z/OS UNIX Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

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

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 may be installed and configured independently on each desktop, or a Java™ Applet version known as Host On-Demand (HOD) can be used to automate the process.

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 z/OS VTAM SNA applications on the z/OS host using Telnet TN3270E, TN3270, 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. The TN3270E Telnet server acts as a VTAM application that activates one VTAM application minor node logical unit (LU) to represent each active TN3270 client connection. Using TN3270 is a two-step process. First the client connects via TCP/IP to the TN3270E Telnet server that is listening on some TCP port (usually port 23). Then the TN3270E Telnet server allocates VTAM resources and establishes a logical session on behalf of the connecting client.

1.1.1 Basic concepts of TN3270

The relationship between the Telnet server, the TN3270 client, and the TCP/IP stack is illustrated in Figure 1-2.

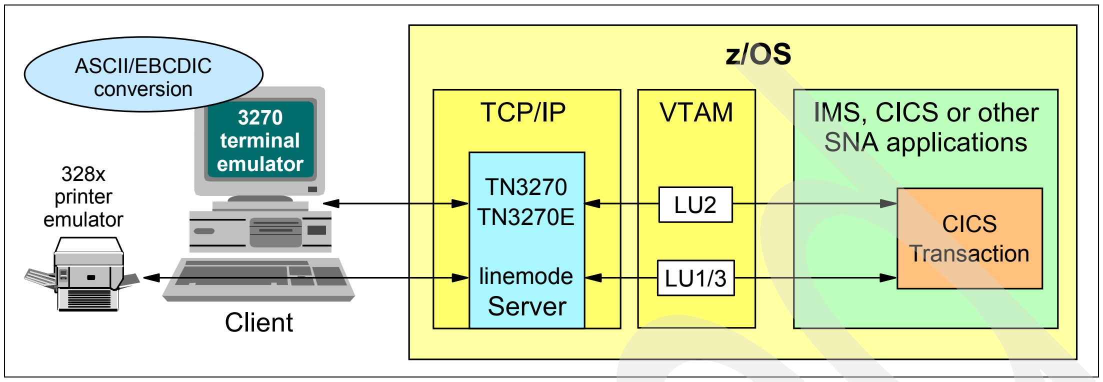


Figure 1-2 TN3270 client/server relationship

Traditional Telnet and TN3270 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 character 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 support includes a number of important enhanced capabilities over TN3270 and are discussed in greater detail in 1.2.1, "TN3270E functionality" on page 4.

1.1.2 Additional information sources

Detailed information for the TN3270E Telnet server and protocol can be found in the following documents:

- *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
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341
- TN3270 is defined by RFC 1646

- TN3270E is defined by RFC 1647 and RFC 2355

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.

1.2 Why TN3270 is important

The TN3270 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.

For quite some time, network devices external to the mainframe have been used to provide TN3270/TN3270E Telnet 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 Telnet server on the mainframe, clients can access it using end-to-end native IP without requiring any SNA network support for their terminal traffic.

Important functions supported by the z/OS TN3270E Telnet server are covered in the following sections:

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

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

1.2.1 TN3270E functionality

Traditional Telnet (line mode, based on ASCII) is very 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 highly recommend that you use TN3270E for all mainframe SNA application connections, unless the client simply does not support it.

Some of the features of TN3270E are illustrated in Figure 1-3 on page 5.

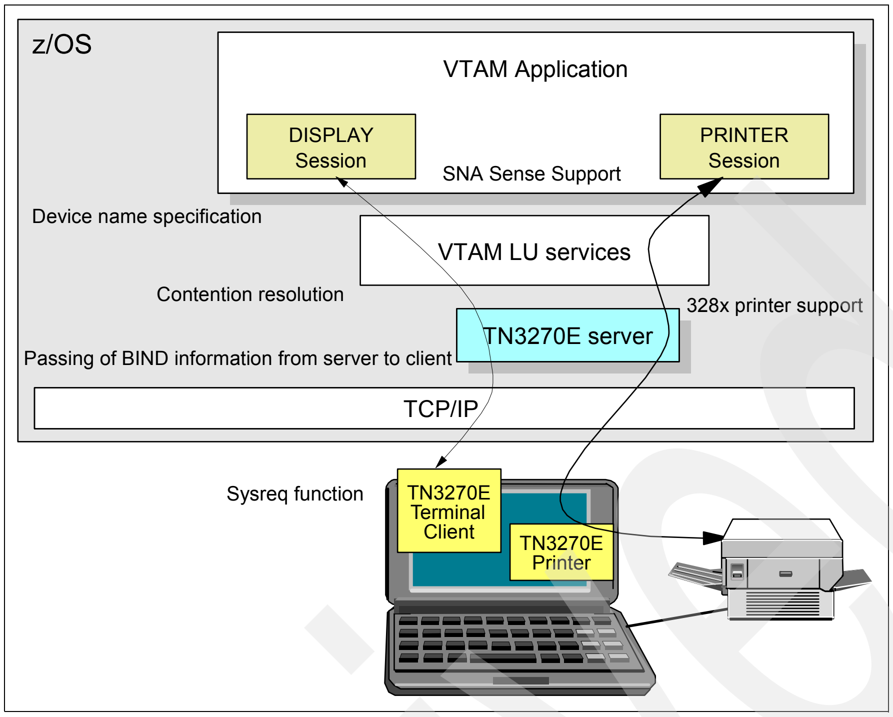


Figure 1-3 TN3270E negotiable features

Once the client and the server have agreed on using TN3270E through negotiation during the connection process, they negotiate the subset of TN3270E options. These options include device-type and a 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

1.2.2 Connection security

Important connection security functions include:

- 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 TN3270 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
- Transport layer security
  - Secures TN3270 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.)

- 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

1.2.3 Connection mapping

When a client connection request is made, TN3270 must assign an LU name to represent the client. Additionally, an Unformated 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. This robust mapping function gives you flexibility in supporting your organization's requirements for running TN3270 services on the z/OS platform.

1.2.4 Connection and session management

In addition to the basic function of facilitating session setup, TN3270 supports several advanced functions such as:

- Session initiation management:
  - Use of VTAM interpret tables
  - Use of VTAM USS tables
  - Use of VTAM default applids
- Connection and session takeover: assists with lost, disconnected sessions.
- 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): TN3270 uses the client certificate to resolve the user ID and RACF® 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.

1.2.5 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), there are several TN3270-specific diagnostic tools available:

- Debug messages: TN3270-specific debug messages can be turned on or off to diagnose TN3270 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 TN3270 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).

1.3 The common design scenarios for TN3270

This section discusses the following TN3270 scenarios:

- Telnet server executing within the TCP/IP address space (In a future release, the Telnet server running in the TCP/IP stack will be planned to discontinue its support.)
- Telnet server executing as a standalone task
- Telnet server with connection security features
- Multiple Telnet servers with SD
- SD with GR applications
- Recommendations for TN3270 configurations

See Appendix B, "Sample files provided with TCP/IP" on page 331, for sample files available for use with the TN3270E Telnet server.

Terminology: In the remainder of this chapter, we will refer to the TN3270E Telnet server as the Telnet server.

1.3.1 Telnet server executing within the TCP/IP address space

This section provides an overview of executing the Telnet server within the TCP/IP stack's address space. This environment is illustrated in Figure 1-4.

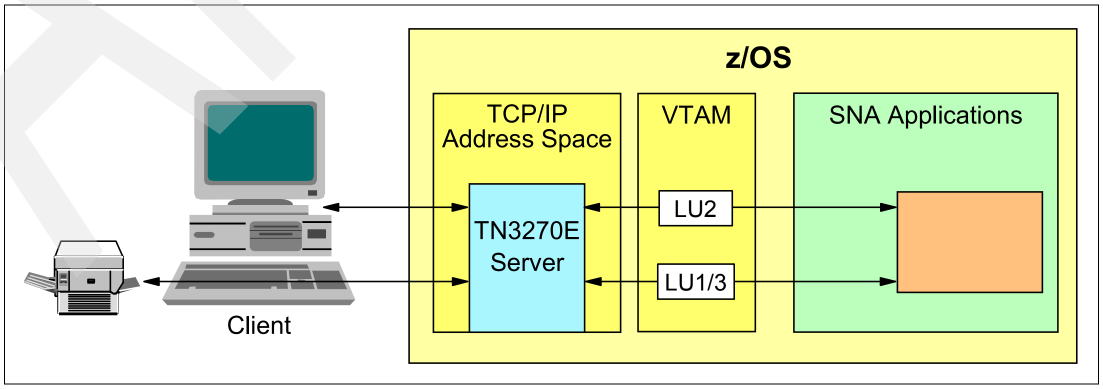


Figure 1-4 Telnet server in the TCP/IP stack's address space

The following topics are discussed:

- Description of Telnet server within the stack's address space
- Dependencies for running TN3270 within stack's address space
- Advantages of running TN3270 within the stack's address space
- Considerations for running TN3270 within the stack's address space

Note: CS for z/OS V1R8 is the last release that will support Telnet server running in the TCP/IP stack. Releases of CS for z/OS beyond V1R8 will have the ability to be notified of RACF terminal class updates, therefore becoming multilevel security compliant.

Description of Telnet server within the stack's address space

For this scenario we use one system image, one TCP/IP stack, and the Telnet server within the stack's address space. Stack affinity is automatically established when the server is defined within the stack's profile configuration data set.

The System used is SC30. The TCP/IP stack started task name is TCPIPB. We use system symbolics in the started task JCL to provide uniqueness where necessary.

Dependencies for running TN3270 within stack's address space

For example, TN3270 must be run within the TCP/IP stack's address space to be multilevel security compliant. The following sections discuss the configuration for running the Telnet server running in the stack's address space.

Common setup activities

Whether TN3270 is running within the TCP/IP stack's address space or in its own address space, both approaches require the following common activities:

- Setting up VTAM LUs for use by TN3270
- Customizing the TN3270 configuration profile data set
- Starting TN3270, either with TCP/IP or in its own address space
- Starting a client emulator and logging on to a SNA application

Port reservation assignment

When TN3270 runs within the TCP/IP stack's address space, a TN3270 port is activated if TN3270 profile statements are specified in the initial profile, or in a data set referenced by a VARY TCPIP,,OBEYFILE command.

If you reserve the TN3270 port when running the Telnet server within the TCP/IP stack's address space, then you must specify INTCLIEN on the port reservation statement.

Stack affinity

The TCPIPJOBNAME parameter in the TELNETGLOBALS block has no meaning and is ignored for TN3270 brought up within the TCP/IP stack's address space. By default, it has affinity to the stack.

Command parameters for VARY TCPIP

Parameters for the VARY TCPIP,,TELNET commands are positional and must be entered in the order shown in the syntax diagrams if running TN3270 as part of the TCP/IP address space. If running TN3270 in its own address space, all parameters after the command can be in any order.

Advantages of running TN3270 within the stack's address space

Since Telnet does not have a mechanism to be notified of RACF terminal class updates, the Telnet server must run within the TCP/IP stack's address space to allow for multilevel security compliant.

Considerations for running TN3270 within the stack's address space

The following are some of the considerations associated with running the Telnet server within the TCP/IP stack's address space:

- Performance

  With both the TCP/IP stack and the Telnet server running within the same address space, they must have the same system performance assignment, and contend for resources on the same level. No granularity can be achieved in setting different performance parameters.

- Availability

  If you are running TN3270 within your TCP/IP stack's address space, an outage (either planned or unplanned) of the Telnet server could adversely affect the whole stack—affecting availability for other TCP/IP applications as well. If the Telnet server is running within your TCP/IP stack, then you will have to recycle the whole stack, thereby affecting the availability of other TCP/IP applications.

1.3.2 Telnet server executing as a standalone task

This section provides an overview of executing the Telnet server as a standalone task in its own address space. This environment is illustrated in Figure 1-5.

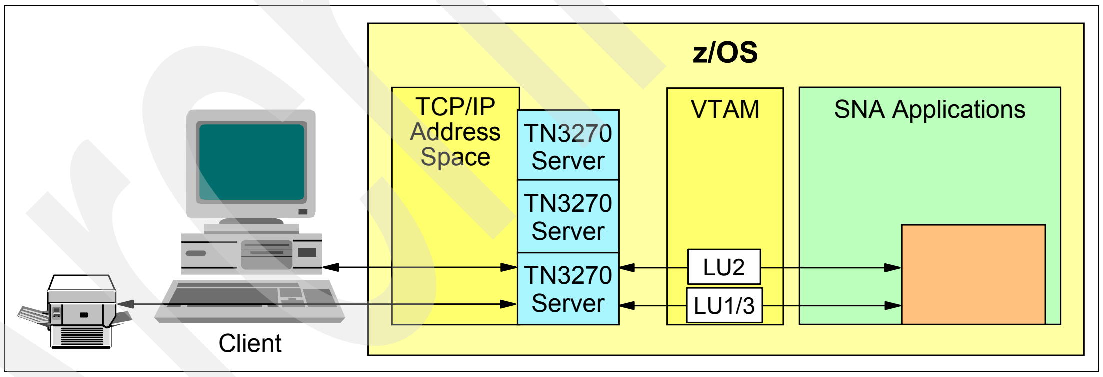


Figure 1-5 Telnet server executing in its own address space

Interestingly (as shown in the diagram), running the Telnet server as a standalone task allows you to run multiple Telnet servers (for example, to support more users than can be supported via a single Telnet server), each in its own address space.

The following topics are discussed:

- Description of Telnet server executing as a standalone task
- Dependencies for Telnet server executing as a standalone task
- Advantages of the Telnet server executing as a standalone task
- Considerations for Telnet server executing as a standalone task

Description of Telnet server executing as a standalone task

TN3270 can be started in its own address space separate from the stack. It has its own JCL procedure defined in a system PROCLIB.

Tip: For migrating your Telnet server out of your TCP/IP stack address space, there is an easy way to prepare the TN3270 profile configuration to facilitate moving it from running within the stack to running external to the stack, as follows.

Place all the TN3270-related statements into a separate configuration member (for example, you could call it PROFTELN), removing all TN3270-related statements from the TCP/IP stack configuration member (such as PROFSTAK).

- When testing TN3270 as a standalone task, point the TN3270 task's //PROFILE DD to the TN3270 profile member, PROFTELN (it contains TN3270-related statements only).
- When you need to run the Telnet server within the stack's address space, simply add an INCLUDE statement for the PROFTELN member into the stack's profile member at an appropriate location. The INCLUDE statement coded in member PROFSTAK would look something like this:

```
INCLUDE SYS1.TCPPARMS(PROFTELN)
```

If you have inserted a TCPIPJOBNAME statement into the TELNETGLOBALS block to achieve stack affinity while running the Telnet server as a standalone task, there is no concern when including the member into the stack because the TCPIPJOBNAME statement is ignored when running within the stack's address space. See the explanation of TCPIPJOBNAME in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Refer to the system diagram in Figure 2 on page xiii as a reference for the following discussions. For this scenario we use one system image, one TCP/IP stack, and one Telnet server running as a separate started task. Stack affinity was established for the Telnet server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block.

System SC30 is used. The TCP/IP stack started task name is TCPIPB. We use system symbolics in the started task JCL to provide uniqueness where necessary. The TN3270 started task name is TN3270B.

On system SC30 we have:

- TCPIPB
- TN3270B

Dependencies for Telnet server executing as a standalone task

The Telnet server running in its own address space has the configuration requirements discussed in the following sections.

Common setup activities

Whether TN3270 is running within the TCP/IP stack's address space or in its own address space, both approaches require the following common activities:

- Setting up VTAM LUs for use by TN3270
- Customizing the TN3270 configuration profile data set
- Starting TN3270, either with TCP/IP or in its own address space
- Starting a client emulator and logging on to a SNA application

Security definitions for the TN3270 started task

When TN3270 executes in its own address space, it is necessary to set up the started task. Before TN3270 can be started, security for the procedure name and its associated user ID must be defined. Refer to the TN3270 chapter in *z/OS Communications Server: IP*

Configuration Guide, SC31-8775, for setting up security for the started task. Also review the sample file SEZAINST(EZARACF) that contains sample security statements for this effort.

Program Properties Table and the AUTOLOG function

The default PPT entry sets the Telnet server to non-cancelable. As a non-cancelable application, the Telnet server should not be started automatically by a TCP/IP stack using the AUTOLOG function. If a TCP/IP stack that has AUTOLOG specified is recycled, the following messages will be 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
```

Command Parameters for VARY TCPIP

Parameters for the VARY TCPIP,,TELNET commands are positional and must be entered in the order shown in the syntax diagrams if running TN3270 as part of the TCP/IP address space. If running TN3270 in its own address space, all parameters after the command can be in any order.

Advantages of the Telnet server executing as a standalone task

Advantages of running TN3270 in its own address space include the following:

- TN3270 failures do not interfere with the integrity of the stack when they are separated.
- TN3270 priority can be set to a different priority than that of TCP/IP.
- TN3270 can be stopped and restarted without stopping TCP/IP.
- Separating TN3270 and TCP/IP makes problem diagnosis easier.
- You can start multiple instances of Telnet servers (for example, for scalability).
- In a common INET environment, TN3270 can be associated with multiple stacks, or have affinity to a single stack by using the TCPIPJOBNAME parameter in the TELNETGLOBALS block.

Considerations for Telnet server executing as a standalone task

Some considerations must be given to running the Telnet server in its own address space.

Command issues

Telnet-related system commands are entered exactly as they are for the Telnet server running within the TCP/IP stack's address space. However, when TN3270 runs in its own address space, the TN3270 procedure name must be specified on all commands instead of the TCP/IP stack procedure name. Otherwise, the command is processed by the default TCP/IP stack instead of the TN3270 procedure. For example, assuming the TN3270 procedure name is TN3270B, the profile display command is as follows:

```
D TCPIP,TN3270B,TELNET,PROFILE          instead of D TCPIP,TCPIPB,TELNET,PROFILE
```

If the procedure does not exist or the procedure name is typed incorrectly, a TCP/IP error message is displayed indicating the procedure is not known.

Note: If you move your Telnet server from running within your stack task to a standalone task you will probably need to update your operational procedures and automation.

Also, when you have a stack task and a standalone TN3270 task running, there are some commands that apply both to the stack and to the Telnet server task. The only way to indicate which task you want the command applied to is to specify the task name.

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 below 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 Telnet 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, including the TN3270 subagent that might be running within the TCP/IP stack's address space.

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 within the TCP/IP stack's address space, the SMF 119 address space name of the writer or the SMF 118 started task name is always the same name, the name of the TCP/IP stack. When running TN3270 as its own procedure, the address space name or started task name is the name of the TN3270 procedure.

Note: If you move your Telnet server from running within your stack task to a standalone task you will probably need to update your SAS, MICS, or other data reductions programs that may be using the started task procedure name to identify records. With the Telnet server within the stack, they are accustomed to seeing the stack task name. When they change to a standalone server task, there will be a new name in the field, and the code may have to be changed.

Port reservation

Reserving a TN3270 port using INTCLIEN does not work when TN3270 is running as its own procedure. INTCLIEN implies that TN3270 is running within the TCP/IP stack's address space. If INTCLIEN is specified, the stack rejects the BIND request from TN3270 running as its own standalone procedure.

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

```
PORT 23 TCP TN3270B NOAUTOLOG
```

Multilevel security compliance

Because the Telnet server does not get notified of RACF terminal class updates, TN3270 running as its own procedure is not multilevel security compliant. Releases of CS for z/OS beyond V1R8 will provide a mechanism to notify the Telnet server of RACF terminal class updates.

1.3.3 Telnet server with connection security features

This section provides an overview of executing the Telnet server with connection security features. This environment is illustrated in Figure 1-6.

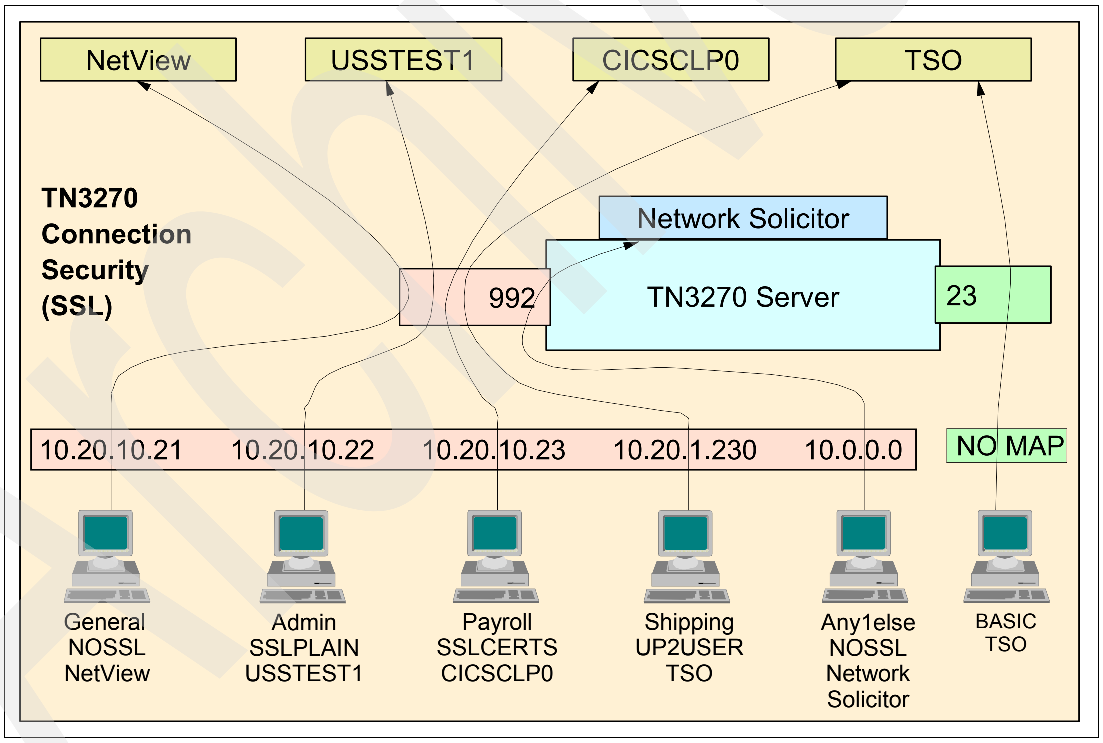


Figure 1-6 Telnet server with connection security (SSL support)

The following topics are discussed:

- Description of Telnet server with connection security
- Dependencies for the Telnet server with connection security
- Advantages of Telnet server with connection security
- Considerations for the Telnet server with connection security

Description of Telnet server with connection security

Telnet server connection security addresses the following security requirements:

- Encrypting traffic between the TN3270 client and server
- Authenticating the server (to ensure that the client is indeed connecting to the server that he expects to be connecting into)
- Authenticating the client (to only provide TN3270 services to specific clients and to customize the service provided to those clients)

To enable Telnet server connection security, the port over which the client connects needs to be configured with the appropriate security parameters.

Note: Most TN3270 security requirements could also be addressed on a stack-wide basis through the use of z/OS Communications Server IP Security (IPSec) and Application-Transparent Transport Layer Security (AT-TLS) capabilities. This book is focused on the implementation of *Telnet server connection security*. IPSec and AT-TLS implementations are discussed in *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

This scenario is set up to run with a standalone TN3270 started task. Only secure mode features are described in this scenario. This scenario builds upon the standalone task scenario, which is:

- Described in "Telnet server executing as a standalone task" on page 9
- Implemented in "Implementing the Telnet server executing as a standalone task" on page 25

For this scenario we use one system image, one TCP/IP stack, and one Telnet server running as a separate started task. Stack affinity has been established for the Telnet server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block.

System SC30 is used. The TCP/IP stack started the task named TCPIPB. We use system symbolics in the started task JCL to provide uniqueness where necessary. The TN3270 started task name is TN3270B.

On system SC30 we have:

- TCPIPB
- TN3270B

We keep the original well-known port 23 defined as a BASIC port with no security support, and make no changes to it.

We add a second port (992) to support our security requirements. We define new Dynamic VIPA (DVIPA) addresses that are to be used by different departments representing various security requirements. We associate these DVIPA addresses with the departments (or user groups) having specific security needs.

We show how one port (992) can be set up to handle a number of client groups with the following security requirements:

- Basic connections (non-secure, general users)
- Required secure connections (any encryption, no client authentication, Admin Dept.)
- Required secure connections (specific encryption, client authentication, Payroll Dept.)
- Optional secure connections (determined by the client, Shipping Dept.)

You can choose to implement your security on one single port as we did, or you can implement a number of secure ports, each with its own set of requirements. The security policies of your organization will dictate the method that is used. You can satisfy those security requirements by carefully choosing the statements you need and where to locate them within the TN3270 profile configuration.

Dependencies for the Telnet server with connection security

To implement secure connections, the Telnet server task must have APF authorized access to the System SSL DLLs. See the *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details.

A key ring file is used to store one or more key pairs and certificates. In order to implement Telnet server connection security, a key ring file is required and must be populated with certificate information before the Telnet server accesses it on behalf of client connection processes.

The Telnet server profile configuration must be updated with appropriate statements supporting the security features selected. There are two essential profile statements required for establishing secure ports:

- SECUREPORT: All TLS/SSL-enabled TN3270 ports must be defined by specifying a TELNETPARMS block for each port. The SECUREPORT port designation statement in the TELNETPARMS block indicates the port is capable of handling SSL connections.
- KEYRING: A server certificate is required for the server authentication process defined by the SSL protocol. This certificate is stored in a key ring. The key ring type and location is specified in the KEYRING statement. Only one key ring can be used by the Telnet server at any given time.

Three other optional, but important, profile statements can be used to assist in defining the types and level of security imposed on the port connections:

- CONNTYPE: The type of secure connection to be used on the port is specified with this statement. If it is specified in the TELNETPARMS block, it sets the default type for the port. If it is specified in a PARMSGROUP block, it can override the port's default setting.
- CLIENTAUTH: The CLIENTAUTH parameter indicates that the client must send a client certificate to the server. If this parameter is not specified, a client certificate is not requested during the SSL handshake and no certificate-based client authentication is done. The level of validation done depends on the option specified.
- ENCRYPTION: The Telnet server has a default ordered list of encryption algorithms it will negotiate with the client, until it synchronizes (negotiates) to one that is compatible with the client. Use this statement to alter the order and to limit the list of supported algorithms. Perhaps you need to limit the choices to the two highest levels of encryption, not allowing the lower levels to even be attempted. If this parameter is not specified, all encryption algorithms that can be specified will be negotiated by the Telnet server.

There are more statements available to customize and secure the ports supported by the Telnet server:

- For a complete list of parameter statements and their syntax, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- For discussions on their use, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Advantages of Telnet server with connection security

Any time there are security concerns for accessing data, connecting to systems, or abusing system resources, security measures must be reviewed and implemented. The Telnet server supports security features that address three main areas of concern: Data Overrun conditions, Transport Layer Security, and Network Access Control. The most commonly used of these is Transport Layer Security (TLS). These three areas are discussed in the following sections:

- Advantages of implementing Data Overrun security
- Advantages of implementing Transport Layer Security (TLS)
- Advantages of implementing Network Access Control

Advantages of implementing Data Overrun security

A number of resource utilization limitations can be established for user connections. These limits can guard such things as:

- The number of bytes received from a client without an End Of Record (EOR) being received
- The number of data segments (RPLs) queued to be sent to VTAM
- The number of session requests received by a TN3270 client in a 10-second period
- The number of chained RUs that can be received over a given session from a host application without a corresponding end chain RU

Advantages of implementing Transport Layer Security (TLS)

The Telnet server provides the ability to secure TN3270 connections with the transport layer security (TLS) or secure sockets layer (SSL) protocol. A port that is configured to use the TLS/SSL protocol is referred to as a *secure port* or SECUREPORT. A port that does not use the TLS/SSL protocol is referred to as a *basic port*.

Connections are also either secure or basic. The flows between the Telnet server and VTAM (and SNA applications) are unaffected by TN3270 security configuration. The Internet Engineering Task Force (IETF) TLS-based Telnet Security Draft, supported by the z/OS Communications Server, allows a TN3270 negotiation to determine whether the client wants or supports TLS/SSL prior to beginning the handshake. The default Telnet server action for a secure port is to first attempt a TLS/SSL handshake. If the client does not start the handshake within the time specified by SSLTIMEOUT, an attempt will be made to negotiate TLS/SSL as defined by the TLS-based Telnet Security Draft. If the client responds that a secure connection is desired, the handshake is started. If the client rejects TLS/SSL, the connection will be closed. This enables installations to support SSL secure clients without knowing ahead of time which protocol the client will use.

The z/OS Communications Server Telnet server supports client authentication. Client authentication is done with the CLIENTAUTH parameter. Client authentication uses RACF services to translate the client certificate to an associated user ID to be used as a client identifier.

Advantages of implementing Network Access Control

Network Access Control (NAC) limits user access to certain IP security zones defined by the NETACCESS statement. A security product, such as RACF, is used to check the permission of user IDs to send data to or receive data from these security zones. The NAC user ID is based on the application's address space information. TCP/IP is an exempt user and has access to all zones. If the Telnet server is running within the stack's address space, then TN3270 processes are also exempt from Network Access Control based on address space user ID information. The NACUSERID parameter enables Network Access Control checking for the Telnet server. This parameter is used to associate Telnet server ports with a user ID defined to the security server. The user ID specified on the NACUSERID parameter must be a valid user ID defined to the security server. If not, the TN3270 port will fail initialization.

Considerations for the Telnet server with connection security

The following are potential issues with the use of TN3270 server connection security:

- The complexity of implementing security policies can delay a normal server implementation.
- Changes to security polices may adversely affect existing methods of connectivity and cause interruption of service.
- Costs associated with additional security measures may not be fully understood or provisioned.
- Implementation of security policies usually crosses departmental boundaries and requires close cooperation and planning. This effort is sometimes underestimated or even overlooked.

Depending on the business requirements of your organization, some type of access security and connection security will be necessary. Some organizations may implement security at the networking level, not requiring any authentication or encryption at the application level. Yet others may dictate that full level 3 digital certificate authentication and SSLV3/TLS encryption be the standard. Therefore, we do not recommend a specific implementation. However, we suggest that you familiarize yourself with the strong security features that TN3270 supports. Refer to the following sites for a better understanding of security methodology:

```
http://www.verisign.com/repository/crptintr.html
http://www.verisign.com/client/about/introCryp.html
```

1.3.4 Multiple Telnet servers with SD

This section provides an overview of executing multiple Telnet servers using Sysplex Distribution (SD) to load balance between them. This environment is illustrated in Figure 1-7.

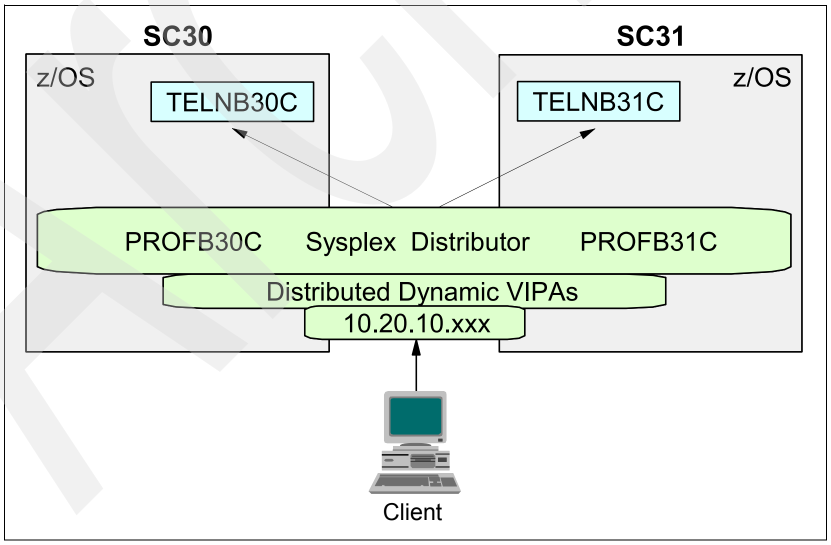


Figure 1-7 Telnet server with sysplex distribution

In Figure 1-7, clients connect to the Distributed DVIPA address of the Telnet server. Based on installation policies, the Sysplex Distributor (SD) will then direct connections to the best available Telnet server.

The following topics are discussed:

- Description of multiple Telnet servers with SD
- Dependencies for multiple Telnet servers with SD
- Advantages of multiple Telnet servers, with SD
- Considerations for multiple Telnet servers, with SD

Description of multiple Telnet servers with SD

This scenario is set up to run with standalone TN3270 procedures; hence, our sysplex distribution environment is built upon the Telnet server standalone task scenario:

- Described in “Telnet server executing as a standalone task” on page 9
- Implemented in “Implementing the Telnet server executing as a standalone task” on page 25

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

For this scenario we use two system images: SC30 and SC31. Each system uses one TCP/IP stack. The TCP/IP stack started task name is the same on both systems: TCPIPB. Each of the stacks has one Telnet server associated with it. The Telnet server started task name is the same on both systems: TN3270B. Stack affinity has been established for the Telnet server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block. We use system symbolics in the started task JCL to provide uniqueness where necessary.

On each system, SC30 and SC31, we have:

- TCPIPB
- TN3270B

We designed the two stacks involved to back each other up. We also designed the two Telnet servers to back each other up. Even though 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 Telnet servers in identical fashion as the primary.

Dependencies for multiple Telnet servers with SD

Dependencies are listed separately for the stacks and for the Telnet servers.

Stack dependencies for multiple Telnet servers with SD

Because sysplex distribution (SD) is being used in this scenario, all the functionality that a TCP/IP stack needs to support SD is required. Refer to *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341, for a discussion on setting up a TCP/IP stack to support sysplex distribution. These requirements include:

- 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 HiperSockets are implemented, HiperSockets use 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 Telnet 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).

Telnet server dependencies for multiple servers with SD

Because the two Telnet servers back each other up, certain parameters must be identically configured so 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, etc.

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

Note: TN3270 LU names to be used with VTAM must be unique for each server.

Advantages of multiple Telnet servers, with SD

Sysplex distribution takes advantage of the existence of multiple, redundant resources to 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 (Telnet servers in this case)

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

For more on the advantages of high availability and workload balancing, refer to those discussions in the following resources:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Considerations for multiple Telnet servers, with SD

Implementing multiple, redundant TCP/IP stacks and Telnet servers increases the effort required of systems personnel in maintaining 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 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. Finally, automation and scheduling changes may be required.

1.3.5 SD with GR applications

This section provides an overview of executing multiple Telnet servers with sysplex distribution providing load balancing between them. They point to an application name that is an active participating member of a VTAM generic resources (GR) group. This environment is illustrated in Figure 1-8.

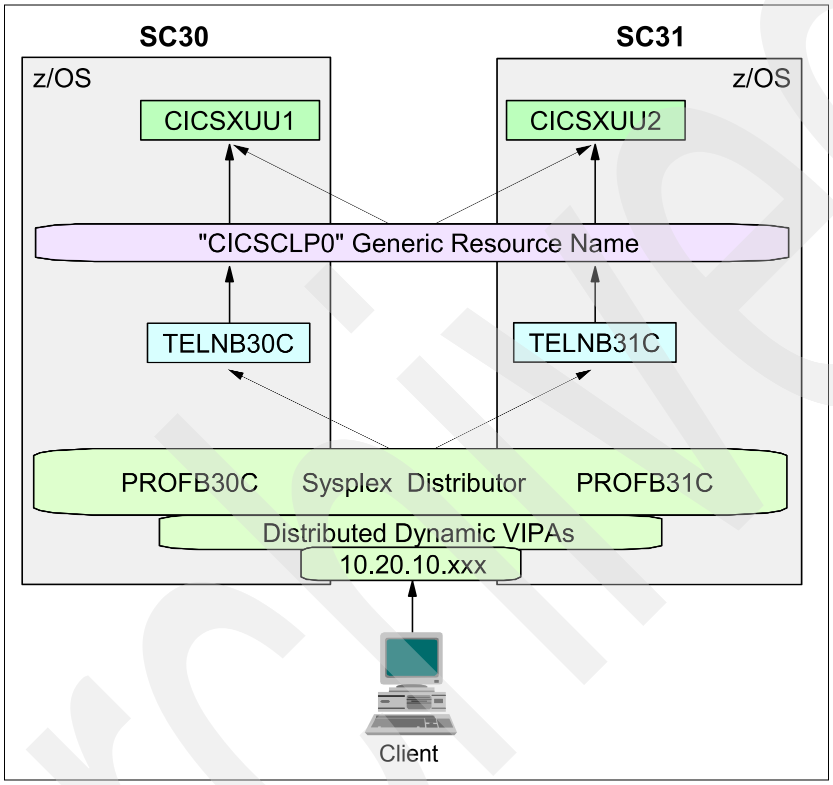


Figure 1-8 Telnet server with sysplex distribution and generic resources applications

The following topics are discussed:

- Description of SD with GR applications
- Dependencies for SD with GR applications
- Advantages of SD with GR applications
- Considerations for SD with GR applications

Description of SD with GR applications

A sysplex distributed environment, distributing connections to multiple Telnet servers, is described in this scenario. In addition, we point the Telnet servers to a generic resource application name that represents two applications that have been set up to run as members of a VTAM generic resources group.

Only the additional required sysplex distribution configuration statements for the stack and for the Telnet server are discussed in this section. We do not discuss how to set up the applications for generic resources support. The implementation of generic resources support is unique to each application and is not within the scope of this book.

This scenario is set up to run with standalone TN3270 procedures. It builds upon the TN3270 standalone task scenario:

- Described in "Telnet server executing as a standalone task" on page 9
- Implemented in "Implementing the Telnet server executing as a standalone task" on page 25

It also builds upon the multiple Telnet servers with sysplex distribution scenario:

- Described in "Multiple Telnet servers with SD" on page 17
- Implemented in "Implementing multiple Telnet servers with SD" on page 72

For this scenario we use two system images: SC30 and SC31. Each system uses one TCP/IP stack. The TCP/IP stack started task name is the same on both systems: TCPIPB. Each of the stacks has one Telnet server associated with it. The Telnet server started task name is the same on both systems: TN3270B. Stack affinity has been established for the Telnet server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block. We use system symbolics in the started task JCL to provide uniqueness where necessary.

The generic resources application information is:

- Generic resource group name is CICSCLP0
- VTAM APPLID of Application 1 is CICSXUU1
- VTAM APPLID of Application 2 is CICSXUU2

On system SC30 we have:

- TCPIPB
- TN3270B
- CICSXUU1

On system SC31 we have:

- TCPIPB
- TN3270B
- CICSXUU2

We designed the two stacks involved to back each other up. We also designed the two Telnet servers to back each other up. Even though 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 continues to distribute connections to the Telnet servers in identical fashion as the primary.

Dependencies for SD with GR applications

In the following sections dependencies are listed for the stacks and for the Telnet servers.

Stack dependencies for SD with GR applications

Because sysplex distribution (SD) is being used in this scenario, all the functionality that a TCP/IP stack needs to support SD is required. Refer to *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341, for a discussion on setting up a TCP/IP stack to support sysplex distribution. These requirements include:

- 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 HiperSockets are implemented, HiperSockets use 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 Telnet 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. 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).

Telnet server dependencies for SD with GR applications

Because the two Telnet servers back each other up, certain parameters must be identically configured so 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, clientids, groups, and parameter overriding statements that affect the assignment of Logmodes, APPLs, LU names, USS tables, Interpret tables, etc.

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

Note: TN3270 LU names to be used with VTAM must be unique for each server.

Advantages of SD with GR applications

In addition to all the advantages listed in “Advantages of multiple Telnet servers, with SD” on page 19, there are advantages associated with running applications under VTAM generic resources. The same high availability that sysplex distribution brings to the TCP/IP applications, generic resources brings to the VTAM applications. Workload balancing does not play as much of a role in generic resources as it does in sysplex distribution, but there is still somewhat of an effort on VTAM’s part to balance the sessions with GR participating applications.

Generic resources is a VTAM and SNA application high-availability function. The purpose for using generic resources is to accomplish high application availability through multiple, redundant instances of the SNA application, running on separate system images within the sysplex. The application instances use a Coupling Facility (CF) structure name called a generic resource group name (STRGR name) with which to associate themselves. They become active participating members of the group when they introduce themselves to VTAM's sysplex services. VTAM uses the CF structure to manage session initiation requests from clients for the GRname. These clients can be VTAM controlled LU terminals, or TN3270 LU names associated with IP TN3270 connections. VTAM attempts to balance the workload among all the active group members by assigning each client logon request to the best choice at the time of logon.

The type of applications can include online transaction applications, such as CICS® and IMS™, or session managers, foreground batch applications such as TSO, or any application that supports generic resources. The use of generic resources especially helps with session managers where TN3270 clients are presented with an application selection menu when they first connect to the system. If the session manager supports generic resources, it can be configured as in the following explanation.

For purposes of this discussion let us assume there are two systems in the sysplex on which we want to implement sysplex distribution support for multiple Telnet servers. Each system has a TCP/IP stack, and each stack has a Telnet server with affinity to the stack. A session manager instance can be configured on each of the two systems. Stack1 on system1, defined as the primary distributing stack, is configured with a Dynamic VIPA that represents the session manager resource. Stack2 on system2, defined as the backup stack, is configured with the same Dynamic VIPA definitions representing the session manager resource. The stacks implement SD functionality to accept TN3270 connection requests and distribute those connections to one of the two Telnet servers in the sysplex.

We must configure the two Telnet servers as identically as possible, treating all client connections the same, differing only in those statements that must be coded with unique values to meet uniqueness requirements imposed by VTAM, TCP/IP, and the sysplex. (TN3270 LU names are an example of this uniqueness requirement.) When the Telnet servers are handed a distributed connection request, they map the connection either directly to the session manager by way of the DEFAULTAPPL statement or to a USS table by way of the USSTCP statement. If a USS table is mapped, then the client can eventually enter a command that will request a logon to the session manager.

Regardless of whether the DEFAULTAPPL is used or the USSTCP is used, the Telnet server forwards eventual requests toward the generic resource group name. VTAM, on the same system as the Telnet server, receives the logon request and, recognizing the request is coming from a TN3270 LU, forwards the logon request to the session manager that is running on that same system. This is VTAM's default behavior. This keeps the TN3270 IP connection and its VTAM LU session contained on the same system, thus avoiding cross-system dependencies. This also avoids the inefficiencies associated with sending client data across system boundaries every time the client presses the Enter key.

Obviously, if a session manager region fails, those users connected to it will lose their sessions. However, in the case of a failed region, VTAM does see the other G/R member still active on the other system, and will route connection requests to it during the absence of the failed region. Therefore, the affected users could immediately reconnect. When the failed region is reintroduced to the GR group, VTAM will once again begin directing new connections to the recovered region. Even though this temporary sending of connections *cross-system* is not the most desirable or efficient method, it does take advantage of the redundant session manager region and keeps client connection requests accommodated during the outage.

Considerations for SD with GR applications

Some considerations must be given to using sysplex distribution with Telnet servers pointing to an application that is a member of a generic resource group.

The additional clerical effort to maintain and synchronize the application configurations and their associated shared database can be quite complex, and requires separate planning and design.

1.3.6 Recommendations for TN3270 configurations

Based on the above discussions, we recommend some scenarios that could meet your TN3270 requirements.

Standalone Telnet server

We recommend running the Telnet server as a standalone procedure in its own address space, not within the TCP/IP stack's address space. Refer to the reasons for doing so in "Advantages of the Telnet server executing as a standalone task" on page 11.

Secure TN3270 connections

Depending on the business requirements of your organization, some type of access security and connection security are probably necessary. Your organization may implement security at the networking level, not requiring any authentication or encryption at the stack or application level. Yet others may dictate that digital certificate authentication and SSLV3/TLS encryption be the standard. We show you how to implement basic TN3270 SSL support both with and without client authentication. We suggest that you familiarize yourself with the strong security features that TN3270 supports.

Note: Most TN3270 security requirements could also be addressed on a stack-wide basis through the use of z/OS Communications Server IP Security (IPSec) and Application-Transparent Transport Layer Security (AT-TLS) capabilities. This book focuses on the implementation of *Telnet server connection security*. IPSec and AT-TLS implementations are discussed in *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

TCP/IP stack redundancy

If you have only one system image or one logical partition (LPAR) to consider, then multiple stacks will not improve availability for you. 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, then we recommend that you implement a TCP/IP stack on each image and take advantage, where possible, of capabilities that enable the multiple stacks to be configured to back up each other.

Telnet 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 may need or want to have more than two, depending on business and technical requirements. You may also want to implement the Telnet server on your other mainframe systems in order to provide direct access to them for systems administration in cases where the centralized systems are not available.

Note: Do not use the same LU names in the multiple Telnet servers.

Sysplex distribution

When you have multiple stacks with multiple Telnet servers, as suggested above, and are leveraging parallel sysplex technology, we recommend that you implement sysplex distribution in order to provide TN3270 connection load balancing at some level: round-robin, basewlm, or serverwlm.

Generic resources (GR)

When your Telnet server associates a connection to an application, whether by DEFAULTAPPL, or a Unformated System Services (USS) table command, or a user-specified `Logon Applid()` command, you will achieve better availability if the target application is itself redundant and has generic resources implemented with VTAM. If the target application is a member of a VTAM generic resource group, then the Telnet server can simply specify the generic resource group name and let VTAM determine which GR member receives the logon request.

1.4 How a TN3270E server is implemented

In the scenarios described above, we documented the situations that require the Telnet server to execute within the TCP/IP stack's address space. If you have to support any of those environments, we understand that you cannot take advantage of executing the Telnet server as a standalone task. Because it is simple to set up the TN3270 profile configuration data set to be used by either the TCP/IP started task or the TN3270 started task, we do not show both scenarios.

Based upon our recommendations in 1.3.6, "Recommendations for TN3270 configurations" on page 23, we prepared implementation details for the scenarios described in 1.3, "The common design scenarios for TN3270" on page 7, except for the Telnet server running within the TCP/IP stack's address space. All of our selected scenarios use a Telnet server executing as a standalone task. We show the implementation details for a standalone task only once, and do not repeat the discussion for the other selected configurations. We use the standalone task example as a base from which to build the others. When reviewing any of the following scenarios, assume the standalone task effort is common for each one.

- Telnet server executing as a standalone task
  - Described in "Telnet server executing as a standalone task" on page 9
  - Implemented in "Implementing the Telnet server executing as a standalone task" on page 25
- Telnet server with connection security features
  - Described in "Telnet server with connection security features" on page 13
  - Implemented in "Implementing the Telnet server with connection security features" on page 58
- Multiple Telnet servers with SD
  - Described in "Multiple Telnet servers with SD" on page 17
  - Implemented in "Implementing multiple Telnet servers with SD" on page 72
- SD with GR applications
  - Described in "SD with GR applications" on page 20
  - Implemented in "Implementing SD with GR applications" on page 87

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 319.
- Sample files for each application are listed in Appendix B, "Sample files provided with TCP/IP" on page 331.

1.4.1 Implementing the Telnet server executing as a standalone task

The Telnet server can execute as a standalone started task.

The following topics discuss 1.4.1, "Implementing the Telnet server executing as a standalone task" on page 25:

- "Implementation tasks for the Telnet server as a standalone task" on page 26
- "Configuration examples for the Telnet server as a standalone task" on page 32
- "Verification steps for the Telnet server as a standalone task" on page 33
- "Managing the Telnet server as a standalone task" on page 43
- "Problem determination for Telnet server as a standalone task" on page 52

Implementation tasks for the Telnet server as a standalone task

The following tasks are necessary to implement the standalone task:

- 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 Telnet server.
- Customize the JCL for the TN3270 started task.
- Start the Telnet server started task.

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

A sample VTAM APPL major node can be found in SEZAINST(IVPLU).

See *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for an in-depth discussion on how to prepare the VTAM LU definitions for the Telnet server. Attention should be given to the values 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. This could create a hang condition for the TN3270 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 a SNA session with TN3270 to avoid CLOSEACB hang conditions.

These Logical Unit (LU) names represent each TN3270 client IP connection. It is this VTAM LU that logs on to a selected VTAM application. The Telnet 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. These APPL definition members must be made available to VTAM by being in the list of data sets specified on the VTAMLST DD statement in the JCL procedure used to start VTAM. This VTAM major node member contains the TN3270 application LUs, and ensures that these LUs will be available for activation after VTAM, TCP/IP, and the Telnet server are started. An APPL statement can be entered for each LU or a model APPL statement can be used. You should use a model statement in order to avoid all the clerical effort of maintaining a large list. There is an extensive discussion on coding techniques for the model APPL statement in *z/OS Communications Server: SNA Resource Definition Reference*, 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. One of the techniques for using system symbolics is shown in Example 1-1.

Example 1-1 Sample APPL model major node

```
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, 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 1-2. The actual name assigned for a connection is determined by the Telnet server mapping rules, and will be assigned at the time of connection negotiation.

Example 1-2 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 automatically activate the application definition major node, include it in ATCCONxx. If multiple Telnet servers are used (for example, multiple TCP/IP stacks in a SD environment), ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail, or the cross-domain session request will fail.

Customize the TCPDATA configuration data set

This file is normally customized during basic stack setup and initialization. For details on customizing the TCPDATA file and the resolver see *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339. TN3270 uses the native MVS™ sockets search order to find a resolver. Neither the resolver_Config environment variable nor the /etc/resolv.conf HFS is used by TN3270 when searching for TCPIP.DATA. If no resolver has been defined, no host name will be 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 Unformated 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. These mapping statements can omit the client ID: DEFAULTAPPL, PRTDEFAULTAPPL, LINEMODEAPPL, USSTCP, and INTERPTCP. There are additional statements whose very names imply a mapping association to the NULL client group: DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTPRT, and DEFAULTPRTSPEC. These uniquely named groups do not require mapping, they imply it.

For complete details on mapping objects to client identifiers, refer to that topic in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. 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. The following discussion assists you with comprehending the text in the books 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 Telnet 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 could turn out to be a clerical effort you do not wish 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 1-3.

Example 1-3 Mapping an individual object or an individual client ID

```
DEFAULTAPPL CICSCLP0 10.20.10.21
LUMAP SC30BB74 USERID,CS06
```

However, groups must be defined prior to referring to them in a mapping statement. This applies to both object groups and client ID groups. We define a few terms in Table 1-1 on page 29.

Table 1-1 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 1-4.

Example 1-4 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, you should 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: Here is another point for consistency: Where the rules allow, use all *group* definitions, even for those single objects and client IDs that you need to map. The mapping statements will then always be referring group names (where the syntax permits), not to a single object or client. Even if the group consists of only one single item, the mapping statements are consistent.

This technique has an additional advantage: if you ever have to add an object or a client ID to the existing single item being mapped, you already have the group 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—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 Telnet 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.

The purpose of the TN3270 configuration statements is to:

1. Define connection characteristics
2. Facilitate session setup with host VTAM applications.
3. 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.

Define system security for the TN3270 started task

This discussion assumes RACF is the security subsystem being used. If another security product is used, refer to its manuals for equivalent setup instructions. When TN3270 executes in its own address space, set up for the started task is necessary. Before TN3270 can be started, security for the procedure name and its associated user ID must be defined. Refer to the TN3270 chapter in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for setting up security for the started task. Also 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 TN3270*.* STDATA(USER(TN3270))
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 would all be 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 TN3270
ALTUSER TN3270 OMVS(UID(0) PROGRAM ('/bin/sh') HOME('/'))
```

In this example the user ID name is TN3270, 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:

- Add the user to RACF:

  ```
  ADDUSER TN3270
  ```

- Permit the user ID:
  - Create a BPX.SUPERUSER FACILITY class profile:

    ```
    RDEFINE FACILITY BPX.SUPERUSER
    ```

  - If this is the first class profile, activate the FACILITY class:

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

  - Permit the user to the class:

    ```
    ALTUSER TN3270 OMVS(UID(23) PROGRAM ('/bin/sh') HOME('/'))
    PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(TN3270) 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.

  - Refresh the FACILITY class:

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

Consider program properties table attributes for the Telnet 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 privileged or system task cause 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 Telnet 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 will be 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

A sample JCL can be found 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.

Start the Telnet server started task

Issue the MVS START command or automate it with an automation package:

```
S TN3270B
```

Configuration examples for the Telnet server as a standalone task

The JCL for the started task is shown in Example 1-5. Notice the use of system symbolics.

Example 1-5 JCL for the Telnet server: TN3270B

```
BROWSE    SYS1.PROCLIB(TN3270B) - 01.00              Line 00000000 Col 001 080
//TN3270B  PROC PARMS='TRC=TN'
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//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(TELNB&SYSCLONE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.)
```

A portion of the TN3270 configuration profile data set is shown in Example 1-6. Refer to Appendix C, “Configuration files: TN3270E standalone started task scenario” on page 339, to review the complete profile.

Example 1-6 TN3270B configuration profile (TELNB30A) for standalone task

```
BROWSE    CS06.TCPPARMS(TELNB30A) - 01.02            Line 00000000 Col 001 080
; ===SC30=========== TN3270E Telnet server Profile for Standalone Task =======-
; No SSL security.  No sysplex distribution in the stack.                -
;        This is a plain server profile, very simple and easy.           -
; ----------------------------------------------------------------------
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
      SC30BB01..SC30BB99
  ENDDEFAULTLUS

  DEFAULTAPPL TSO           ; All users go to TSO
  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 used for this scenario, see Appendix C, “Configuration files: TN3270E standalone started task scenario” on page 339.

Verification steps for the Telnet server as a standalone task

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.

Important: When a VARY TCPIP,,OBEYFILE command is issued, TELNETPARMS and BEGINVTAM blocks are both required for each port started or modified.

The Telnet server environment can be verified as follows:

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

Using OBEYFILE to modify a configuration

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

Important: When using the VARY TCPIP,,OBEYFILE command to update the TN3270 configuration, new profile statements *completely replace the profile statements* that were 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. The structural layout of the profiles and how connections are associated with profiles is shown in the following figure.
- 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 have ended, the storage for the non-current profile mapping rules is freed and the profile is considered INACTIVE.

Using 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 1-7.

Example 1-7 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 1-8.

Example 1-8 OBEYFILE command to validate or activate a TN3270 profile

```
V TCPIP,TCPIPB,OBEYFILE,TCPIPB.TCPPARMS(TELNB30A)
```

Example 1-9 shows the resulting messages from the OBEYFILE command when the TESTMODE statement is *included* in the TN3270 profile.

Example 1-9 Messages issued when TESTMODE is included in the TN3270 profile

```
V TCPIP,TCPIPB,OBEYFILE,TCPIPB.TCPPARMS(TELNB30A)
   EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPB,OBEYFILE,TCPIPB.TCPPARMS(TELNB30A)
   EZZ0300I OPENED OBEYFILE FILE 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ0400I TELNET/VTAM (SECOND PASS) BEGINNING FOR FILE: 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ6018I TELNET PROFILE TESTMODE COMPLETE FOR PORT    23
EZZ0403I TELNET/VTAM (SECOND PASS) COMPLETE FOR FILE: //'TCPIPB.TCPPARMS(TELNB30A)'
```

Example 1-10 shows the resulting messages from the OBEYFILE command when the TESTMODE statement was *removed* from the TN3270 profile.

Example 1-10 Messages issued when TESTMODE is removed from the TN3270 profile

```
V TCPIP,TCPIPB,OBEYFILE,TCPIPB.TCPPARMS(TELNB30A)
   EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPB,OBEYFILE,TCPIPB.TCPPARMS(TELNB30A)
   EZZ0300I OPENED OBEYFILE FILE 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ0400I TELNET/VTAM (SECOND PASS) BEGINNING FOR FILE: 'TCPIPB.TCPPARMS(TELNB30A)'
   EZZ6003I TELNET LISTENING ON PORT    23
   EZZ0403I TELNET/VTAM (SECOND PASS) COMPLETE FOR FILE: //'TCPIPB.TCPPARMS(TELNB30A)'
```

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

Important: What you specify as the started task name in the `VARY TCPIP,`*`stcname`*`,OBEYFILE` command determines where the new profile will be established. If you specify the stack's stcname, the new profile will be made effective in the stack's address space, whether or not you already have a TN3270 profile active in the stack. Make sure this is where you intend for the new profile to take effect. If you specify the Telnet server's stcname, the new profile will be made effective in the server's address space. If there is no Telnet server task running to be specified in the command, obviously you cannot implement the "first" standalone server task using the OBEYFILE command. In this case, you would use the stack's stcname for the TESTMODE syntax check, and then start the Telnet server task, referencing the new profile. Once you have an active Telnet server task, it can be specified in the OBEYFILE command to process new profile scans and activations.

Checking maximum connections supported

For each port, TN3270 uses setrlimit() to automatically set the MaxFileProc value to the maximum allowed by UNIX® System Services, 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.

Checking 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.

Using DISPLAY commands to view the status of TN3270 resources

For all commands referred to in this section, see:

- *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,,TELNET system command to request TELNET information. When running TN3270 in its own address space, the TN3270 procedure name must be specified.

If the stack is running in IPv6 or 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 an 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,,TELNET,PROF,....

- D TCPIP,,TELNET,CLID,....
- D TCPIP,,TELNET,OBJ,....
- D TCPIP,,TELNET,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. Once a match is found, the other parameters are used to determine what is displayed for the profile.

Using 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 in Example 1-11.

Example 1-11 Display PROF, SUM shows profile summary for all profiles

```
D TCPIP,TN3270B,T,PROF,SUM
   EZZ6060I TELNET PROFILE DISPLAY 052
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD*
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      0
   ------------------------------------------------------------
      FORMAT           LONG
      TCPIPJOBNAME     TCPIPB
      TNSACONFIG       DISABLED
   6 OF 6 RECORDS DISPLAYED
```

A profile detail report shows the port settings and includes a legend to assist in interpreting the abbreviated settings, as seen in Example 1-12.

Example 1-12 Display PROF, DET shows profile detail for all profiles

```
D TCPIP,TN3270B,T,PROF,DET
   EZZ6080I TELNET PROFILE DISPLAY 054
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     *******  **TSBTQ***T  EC*  BB*******  ***STS  *DD* *DEFAULT
     -------  ----------T  ---  ---------  ------  ---- *TGLOBAL
     LM-----  ---S-------  --F  -B-------  *--ST-  S--- *TPARMS
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD* CURR
    PERSISTENCE
      LUSESSIONPEND
      MSG07
      ....
    FUNCTIONS
      TN3270E
      SNAEXTENT
      ....
    DIAGNOSTICS
       DEBUG             EXCEPTION
       DEBUG ROUTING     CONSOLE
       FULLDATATRACE
   SECURITY
       PORT                   23
       CONNTYPE          BASIC
       KEYRING           **N/A**
       CRLLDAPSERVER     **N/A**
       ENCRYPTION        **N/A**
       CLIENTAUTH        **N/A**
       NOEXPRESSLOGON
       NONACUSERID
       NOSSLV2
     TIMERS
       .....
   MISCELLANEOUS
     ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      0
     ------------------------------------------------------------
       FORMAT            LONG
       TCPIPJOBNAME      TCPIPB
       TNSACONFIG        DISABLED
    88 OF 88 RECORDS DISPLAYED
```

The profile report can be limited to BASIC ports only, as seen in Example 1-13.

Example 1-13 Display PROF, BASIC, SUM shows profile summary for basic profiles

```
D TCPIP,TN3270B,T,PROF,PROF=BASIC,SUM
   EZZ6060I TELNET PROFILE DISPLAY 056
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD*
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      0
   ------------------------------------------------------------
      FORMAT            LONG
      TCPIPJOBNAME      TCPIPB
      TNSACONFIG        DISABLED
   6 OF 6 RECORDS DISPLAYED
```

The profile report can be limited to SECURE ports only, as seen in Example 1-14. Because there are no secure ports defined in this profile, message EZZ6057I indicates there are no entries to list.

Example 1-14 Display PROF, SECURE, SUM shows profile summary for secure profiles

```
D TCPIP,TN3270B,T,PROF,PROF=SECURE,SUM
   EZZ6060I TELNET PROFILE DISPLAY 060
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

Using 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 may have defined, as seen in Example 1-15.

Example 1-15 Display CLID, SUM shows clientid summary for all profiles

```
D TCPIP,TN3270B,T,CLID,SUM
   D TCPIP,TN3270B,T,CLID,SUM
   EZZ6082I TELNET CLIENTID LIST 071
   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
   ------------------------------------------------------------
   24 OF 24 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 seen in Example 1-16.

Example 1-16 Display CLID, DET shows clientid detail for all profiles

```
D TCPIP,TN3270B,T,CLID,DET
   EZZ6081I TELNET CLIENTID DISPLAY 076
   CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
   NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
   ------------------ ------ --------- -------- ---------- --------
   NULL
     NULL
                           1 DEFAPPL   TSO                 --------
   ----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   5 OF 5 RECORDS DISPLAYED
```

The ClientID report can be limited to BASIC ports only, as seen in Example 1-17 on page 39.

Example 1-17 Display CLID, BASIC, SUM shows client ID summary for basic profiles

```
D TCPIP,TN3270B,T,CLID,PROF=BASIC,SUM
   EZZ6082I TELNET CLIENTID LIST 078
   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
   ------------------------------------------------------------
```

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 seen in Example 1-18.

Example 1-18 Display CLID, SECURE, SUM shows client ID summary for secure profiles

```
D TCPIP,TN3270B,T,CLID,PROF=SECURE,SUM
   EZZ6081I TELNET CLIENTID DISPLAY 082
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

Using 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 may be defined in the profile and what their assignments are, as seen in Example 1-19.

Example 1-19 Display OBJ, SUM shows object summary for all profiles

```
D TCPIP,TN3270B,T,OBJ,SUM
   EZZ6084I TELNET OBJECT LIST 084
   ARAPPL
     SC30N*    NVAS*     TSO*      *
   DEFAPPL
     TSO
   PRTAPPL
```

```
  NO OBJECTS
LINEAPPL
  NO OBJECTS
MAPAPPL
  NO OBJECTS
USS
  NO OBJECTS
INT
  NO OBJECTS
LU
  NO OBJECTS
LUGRP
  *DEFLUS*
APPLLUG
  NO OBJECTS
PRT
  NO OBJECTS
PRTGRP
  NO OBJECTS
PARMSGRP
  *DEFAULT  *TGLOBAL  *TPARMS
MONGRP
  NO OBJECTS
----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
------------------------------------------------------------
30 OF 30 RECORDS DISPLAYED
```

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

The object detail report indicates how many connections there are for each object group, as seen in Example 1-20.

Example 1-20 Display OBJ, DET shows object detail for all profiles

```
D TCPIP,TN3270B,T,OBJ,DET
   EZZ6083I TELNET OBJECT DISPLAY 086
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   ARAPPL
    SC30N*          0
                                                            -A------
    NVAS*           0
                                                            -A--Q---
    TSO*            0
                                                            -A-D----
    *               1
                                                            -A------
    *DEFAPPL        0
                                                            -A------
   DEFAPPL
    TSO             1 NULL      NULL
                                                            --------
   LUGRP
    *DEFLUS*        1
                                                            --------
   PARMSGRP
    *DEFAULT          -------NO MAPPING---------
                                                            --------
    *TGLOBAL         -------NO MAPPING---------
                                                         --------
    *TPARMS          -------NO MAPPING---------
                                                         --------
   ----- PORT:    23  ACTIVE             PROF: CURR CONNS:     1
   ------------------------------------------------------------
   26 OF 26 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 seen in Example 1-21.

Example 1-21 Display OBJ, SECURE, SUM shows object summary for secure profiles

```
D TCPIP,TN3270B,T,OBJ,PROF=SECURE,SUM
   EZZ6083I TELNET OBJECT DISPLAY 092
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

Using 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 seen in Example 1-22.

Example 1-22 Display CONN shows all connections to the Telnet server

```
D TCPIP,TN3270B,T,CONN,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 094
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   000000E8    ::FFFF:10.20.2.242..1028
                                   SC30BB01 SC30TS05  TA3 SNX32704
   ----- PORT:    23  ACTIVE             PROF: CURR CONNS:     1
   ------------------------------------------------------------
4 OF 4 RECORDS DISPLAYED
```

The connection report can be limited to basic ports only, showing only basic connections, as seen in Example 1-23.

Example 1-23 Display CONN, BASIC shows the basic connections only

```
D TCPIP,TN3270B,T,CONN,PROF=BASIC,MAX=*
   D TCPIP,TN3270B,T,CONN,PROF=BASIC
   EZZ6064I TELNET CONNECTION DISPLAY 096
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   000000E8    ::FFFF:10.20.2.242..1028
                                   SC30BB01 SC30TS05  TA3 SNX32704
   ----- PORT:    23  ACTIVE             PROF: CURR CONNS:     1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The connection report shows secure ports and secure connections, as shown in Example 1-24 on page 42.

Example 1-24 Display CONN, SECURE shows the secure connections

```
D TCPIP,TN3270B,T,CONN,PROF=SECURE,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 098
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and SUM parameter gives a *summary* look at one single connection, as seen in Example 1-25.

Example 1-25 Display CONN, CONN=, SUM shows summary information for one connection only

```
D TCPIP,TN3270B,T,CONN,CONN=000000E8,SUM
   EZZ6064I TELNET CONNECTION DISPLAY 100
            EN                                            TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   000000E8    ::FFFF:10.20.2.242..1028
                                     SC30BB01 SC30TS05  TA3 SNX32704
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and DET parameter gives a *complete* look at one single connection, as seen in Example 1-26.

Example 1-26 Display CONN, CONN=, DET shows detail information for one connection only

```
D TCPIP,TN3270B,T,CONN,CONN=000000E8,DET
   EZZ6065I TELNET CONNECTION DISPLAY 102
     CONNECTED: 23:13:55  08/21/2006  STATUS: SESSION ACTIVE
     CLIENT IDENTIFIER FOR CONN: 000000E8   SECLABEL: **N/A**
       CLIENTAUTH USERID: **N/A**
       HOSTNAME: NO HOSTNAME
       CLNTIP..PORT: ::FFFF:10.20.2.242..1028
       DESTIP..PORT: ::FFFF:10.20.1.230..23
       LINKNAME: STAVIPA1LNK
     PORT:    23 QUAL: NONE
       AFFINITY: TCPIPB
       STATUS: ACTIVE  BASIC         ACCESS: NON-SECURE
     PROTOCOL: TN3270           DEVICETYPE: IBM-3278-4-E
       TYPE: TERMINAL GENERIC
       OPTIONS: -TET----   3270E FUNCTIONS: *N/A*
                          NEWENV FUNCTIONS: --
     LUNAME: SC30BB01
     APPL: SC30TS05
       USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
       LOGMODES  TN REQUESTED: SNX32704  APPL SPECIFIED: SNX32704
     MAPPING TYPE:  CONN IDENTIFIER
                         OBJECT   ITEM SPECIFIC     OPTIONS
                   NL (NULL)
                        >*DEFLUS*                   --------
       DEFLT APPL: NL (NULL)
                         TSO                        --------
       USS TABLE:  **N/A**
       INT TABLE:  **N/A**
       PARMS:
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     *******  **TSBTQ***T  EC*  BB*******  ***STS  *DD* *DEFAULT
     -------  ----------T  ---  ---------  ------  ---- *TGLOBAL
     LM-----  ---S-------  --F  -B-------  *--ST-  S--- *TPARMS
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD* TP-CURR
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD* <-FINAL
   35 OF 35 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 seen in Example 1-27.

Example 1-27 Display CONN, LUNAME=, SUM shows summary information for one LU name only

```
D TCPIP,TN3270B,T,CONN,LUNAME=SC30BB01,SUM
   EZZ6064I TELNET CONNECTION DISPLAY 104
            EN                                           TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- --------------------- -------- --------  --- --------
   000000E8    ::FFFF:10.20.2.242..1028
                                      SC30BB01 SC30TS05  TA3 SNX32704
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 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.

Managing the Telnet server as a standalone task

The following commands can be useful when managing the Telnet server environment:

- Using VARY to quiesce, resume, and stop a TN3270 port
- Using VARY to change the status of TN3270 LUs

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

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

- VARY TCPIP,,TELNET,QUIESCE a port to block any new connection requests but allow existing connections to continue activity.
- VARY TCPIP,,TELNET,RESUME a port to end the QUIESCEd state and allow new connection requests.
- VARY TCPIP,,TELNET,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,,OBEYFILE, to start, restart, or change a port by updating the TN3270 profile.

The VARY TCPIP,,TELNET,STOP and VARY TCPIP,,OBEYFILE commands can be used to stop a TN3270 port and then restart that port or a new port without stopping the TCP/IP stack.

The QUIESCE command is shown in Example 1-28.

Example 1-28 QUIESCE Port 23

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

To verify the quiesce command, a display profile command is shown in Example 1-29.

Example 1-29 Port status after QUIESCE command

```
D TCPIP,TN3270B,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 114
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD*
   ----- PORT:    23  QUIESCED             PROF: CURR CONNS:      1
   ------------------------------------------------------------
      FORMAT           LONG
      TCPIPJOBNAME     TCPIPB
      TNSACONFIG       DISABLED
   6 OF 6 RECORDS DISPLAYED
```

If a client attempts to connect while the port is quiesced, the request is rejected, as shown in Example 1-30.

Example 1-30 Connection request is rejected when port is quiesced

```
A TSO telnet command was directed to this port:
   TELNET 10.20.1.230
   EZA8200I MVS TCP/IP TELNET CS V1R8
   EZA8256I Connecting to 10.20.1.230, port TELNET (23)
    EZA8262I Foreign host rejected the open attempt (8547)
    ***
```

The RESUME command is shown in Example 1-31.

Example 1-31 RESUME port 23

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

To verify the resume command, a display profile command is shown in Example 1-32.

Example 1-32 Port status after RESUME command

```
D TCPIP,TN3270B,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 141
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD*
   ----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
      FORMAT           LONG
      TCPIPJOBNAME     TCPIPB
      TNSACONFIG       DISABLED
   6 OF 6 RECORDS DISPLAYED
```

An example of the STOP command is shown in Example 1-33 on page 45. 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 1-33 STOP port 23

```
V TCPIP,TN3270B,T,STOP,PORT=23
   EZZ6038I TELNET COMMAND STOP 23 COMPLETE
   IKT100I USERID          CANCELED DUE TO UNCONDITIONAL LOGOFF
   IKT122I IPADDR..PORT 10.20.2.242..1028
EZZ6010I TELNET SERVER ENDED FOR PORT    23
```

To verify the stop command, a display profile command is shown in Example 1-34.

Example 1-34 Port status after STOP command

```
D TCPIP,TN3270B,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 358
   EZZ6057I NO QUALIFYING MATCHES
0 OF 0 RECORDS DISPLAYED
```

Using the OBEYFILE command to restart port 23 is shown in Example 1-35.

Example 1-35 Restart port 23 with OBEYFILE command

```
V TCPIP,TN3270B,OBEYFILE,TCPIPB.TCPPARMS(TELNB30A)
   EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 476
               TCPIPB.TCPPARMS(TELNB30A)
   EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 477
               TCPIPB.TCPPARMS(TELNB30A)
   EZZ6003I TELNET LISTENING ON PORT    23
   EZZ6038I TELNET COMMAND OBEYFILE  COMPLETE
```

To verify that port 23 is once again defined and active, a display profile command is shown in Example 1-36.

Example 1-36 Port status after restart with OBEYFILE command

```
D TCPIP,TN3270B,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 571
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     LM*****  **TSBTQ***T  ECF  BB*******  ***STS  SDD*
   ----- PORT:    23  ACTIVE             PROF: CURR CONNS:      0
   ------------------------------------------------------------
      FORMAT           LONG
      TCPIPJOBNAME     TCPIPB
      TNSACONFIG       DISABLED
   6 OF 6 RECORDS DISPLAYED
```

Using VARY to change the status of TN3270 LUs

VARY TCPIP,,TELNET,ACT and VARY TCPIP,,TELNET,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 1-37.

Example 1-37 Display inactive TN3270 LUs before an INACT

```
D TCPIP,TN3270B,T,INACTLUS
```

```
   EZZ6061I TELNET INACTLUS DISPLAY 578
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

An example of the INACT command to inactivate a TN3270 LU is shown in Example 1-38.

Example 1-38 INACT command to inactivate a TN3270 LU

```
V TCPIP,TN3270B,T,INACT,SC30BB26
   EZZ6038I TELNET COMMAND INACT SC30BB26 COMPLETE
```

To show a list of inactive TN3270 LUs, use the Display INACTLUS, command as shown in Example 1-39.

Example 1-39 Display inactive TN3270 LUs after an INACT

```
D TCPIP,TN3270B,T,INACTLUS
   EZZ6061I TELNET INACTLUS DISPLAY 582
   INACTIVE LUS
                SC30BB26
   1 OF 1 RECORDS DISPLAYED
```

The ACT command to activate a TN3270 LU is shown in Example 1-40.

Example 1-40 ACT command to activate a TN3270 LU

```
V TCPIP,TN3270B,T,ACT,SC30BB26
   EZZ6038I TELNET COMMAND ACT SC30BB26 COMPLETE
```

To show a list of inactive TN3270 LUs, use the Display INACTLUS command, as shown in Example 1-41. Because we reactivated SC30BB33, it no longer shows as an inactive LU.

Example 1-41 Display inactive TN3270 LUs after an ACT

```
D TCPIP,TN3270B,T,INACTLUS
   EZZ6061I TELNET INACTLUS DISPLAY 605
   INACTIVE LUS
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 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 will check the existing connections to make sure they are still up. If not, the previously existing connections are cleaned up immediately. This improves the situation in which a user thinks his session has failed and he starts a new session (see Figure 1-9 on page 47).

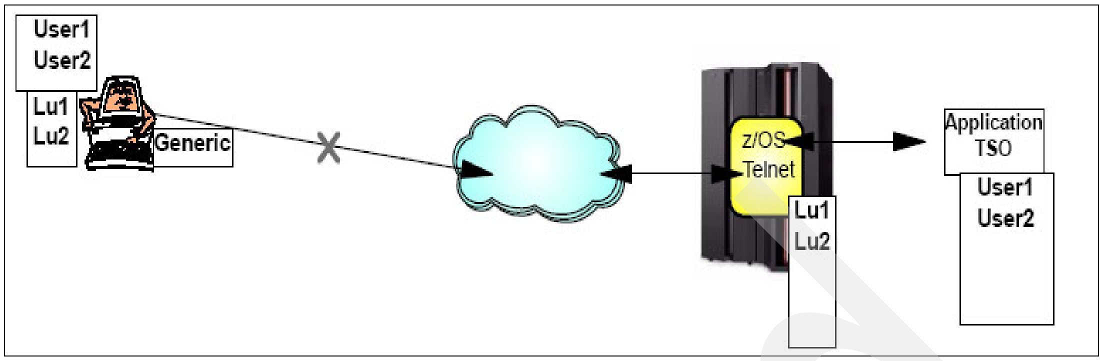


Figure 1-9 Client connection status

The *check client connection* function sends a TIMEMARK value to every preexisting connection 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 very 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/Timemark, creates high overhead. The check client connection method has much less overhead.

Important: In order to have this function enabled in you system, you must have your Telnet server running in a separate address space.

The new statements **CheckClientConn** and **NoCheckClientConn** has the following syntax:

```
CheckClientConn sec,maxconn
```

where *sec* specifies the number of seconds telnet will wait 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 excluded connections with parmsgroup/parmsmap statements.

Tip: This parameter can be 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 may be needed in this situation.

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

Example 1-42 Example showing the definition for clients with the maximum of 60 telnet connections.

```
TelnetParms
  Port 23
Inactive 0
PrtInactive 0
TimeMark 600
ScanInterval 120
CheckClientConn 5,60
EndTelnetParms
```

To be sure about the definitions made in the Telnet profile, use the command in Example 1-43 on page 48.

Example 1-43 Telnet profile without CheckClientConn enabled.

```
D TCPIP,TN3270C1,TELNET,PROF,DETAIL
EZZ6080I TELNET PROFILE DISPLAY 714
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS  MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
  -------  ------------  ---  ---------  ------  ----
  *******  **TSBTQ***RT  EC*  BB*******  ***STS  *DD* *DEFAULT
  -------  ------------  ---  ---------  ------  ---- *TGLOBAL
  LM-----  --------W--T  ---  -B-------  *-*ST-  --L- *TPARMS
  LM*****  **TSBTQ*W*RT  EC*  BB*******  ***STS  *DL* CURR
 PERSISTENCE
   LUSESSIONPEND
   MSG07
   NOTKOSPECLU
   NOTKOGENLU
   NOCHECKCLIENTCONN
   NODROPASSOCPRINTER
   KEEPLU                 0 (OFF)
 FUNCTIONS
   NOOLDSOLICITOR
   NOSINGLEATTN
   TN3270E
   SNAEXTENT
   UNLOCKKEYBOARD BEFOREREAD
   UNLOCKKEYBOARD TN3270BIND
   SEQUENTIALLU
   NOSIMCLIENTLU
```

To compare the before and after the definition changes, reissue the command (see Example 1-44).

Example 1-44 Telnet profile with the CheckClientConn enabled.

```
D TCPIP,TN3270C1,TELNET,PROF,DETAIL
EZZ6080I TELNET PROFILE DISPLAY 663
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS  MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
  -------  ------------  ---  ---------  ------  ----
  *******  **TSBTQ***RT  EC*  BB*******  ***STS  *DD* *DEFAULT
  -------  ------------  ---  ---------  ------  ---- *TGLOBAL
  LM--C--  --------W--T  ---  -B-------  *-*ST-  --L- *TPARMS    1
  LM**C**  **TSBTQ*W*RT  EC*  BB*******  ***STS  *DL* CURR
 PERSISTENCE
   LUSESSIONPEND
   MSG07
   NOTKOSPECLU
   NOTKOGENLU
   CHECKCLIENTCONN        5,60
   NODROPASSOCPRINTER
   KEEPLU                 0 (OFF)
 FUNCTIONS
   NOOLDSOLICITOR
   NOSINGLEATTN
   TN3270E
   SNAEXTENT
   UNLOCKKEYBOARD BEFOREREAD
   UNLOCKKEYBOARD TN3270BIND
   SEQUENTIALLU
   NOSIMCLIENTLU
   WLMCLUSTERNAME
```

1 Indicates that the parameter was set in the TelnetParms statement.

Attention: This function should be used in place of setting low Scaninterval/Timemark values. ScanInterval/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 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 may 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.

Attention: A USS 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 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 USS tables

In recent releases it has been possible to use symbolic variables, such as luname, IP addresses, IP port, and hostname in the MSG10 definition. 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 1-45 illustrates usage of system symbols.

Example 1-45 USS Table Source code with SYSNAME and SYSR1

```
MSG10    DC      AL2(MSG10E-MSG10S)
```

```
MSG10S   EQU     *
         DC      X'F5C31140401D40' ERASE/WRITE,WCC,SBA R1C1,
 DC C'WTSC59 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: &&SYSR1.        '
 DC CL50'                                                 '
 DC CL30'                              '
```

Attention: Note the extra ‘&’ on the symbolic name. This is necessary.

This function can be very useful for diagnostic information; for example, including the LPAR name can be very 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.

In Example 1-46 we can see the definitions that we used in Example 1-45.

Example 1-46 The USSTEST2 compiled including SYSNAME and SYSR1

```
WTSC59 You are connected to a non-SNA terminal - - -

 System Name:  SC30                           z/OS Release: Z18RE1

                                @@@                 @@@  @@@@@@@@@
                           @@@@@@     @@@@         @@@ @@@@@@@@@@@@@@@@@   Inter
national Technical       @@@@@@@@@@@@@@@          @@@@@@@@@@@@@@@@@@@@@    Suppo
rt Organization (ITSO)    @@@@@@@@  @@@        @ @ @ @@@@@@@@@@@@@@@@@@
                           @@@@@@@@@@@           @@@@@@@@@@@@@@@@@@@@@@    Pough
keepsie Center               @@@@@@@@@          @@@@@@@@@@@@@@@@@@@@@@@
                             @ @@@@  @         @@   @ @@  @@@@@@@@@@@ @@
                                @@             @@@@@@ @@@@@@@@@@@@@@  @@
```

The corresponding definition in the TCP/IP profile could be as shown in Example 1-47.

Example 1-47 TCP/IP profile mapping the new USS Table

```
USSTCP USSTEST1              1
USSTCP USSTEST2 9.12.4.197   2
```

– 1 USS Table for general purposes;

– 2 USS Table mapped for an specific IP address.

Queued session timer

Logon manager applications are very popular and usually operate as a default application that sends a selection screen to the end user. Once the end 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 end 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 1-48 Display to see options for Allow/Restrict Appls

```
D TCPIP,TN3270C1,T,OBJ,TYPE=ARAPPL
EZZ6083I TELNET OBJECT DISPLAY 504
OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
----------  ------ --------- ---------------- ---------- --------
ARAPPL
 SC*             0
                                                          -A------  1
                                                        5 ----Q---  2
 TSO*            0
                                                          -A------
                                                        5 ----Q---
 *               0
                                                          -A------
                                                        5 ----Q---
DEFAPPL          0
                                                          -I------
----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
------------------------------------------------------------
14 OF 14 RECORDS DISPLAYED
```

In Example 1-48 we can see the following:

- 1 The 'A' in the OPTIONS field indicates AllowAppl;
- 2 The 'Q' in the OPTIONS field indicates QSession;
- 2 The '5' in the ITEM SPECIFIC field is the time value.

Important: Some considerations are important when using a Qsession Timer:

- Atimer 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 end user appears to see a hung terminal and manually terminates the session.

Performance Monitoring Data Collection

Telnet server performance data was difficult to collect in earlier 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 Telnet 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.

Customers should be encouraged 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 via Telnet SMF SNA session termination record (Type 119/subtype 21). The following data 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 parameters **`MonitorGroup`** and **`MonitorMap`** in the BeginVtam block must be in place for Telnet to capture performance data.

Additional information about the Network Management Interface is available in the *IP Programmer's Guide and Reference* (Chapter 13 Network management)*. S*tructures EZBNMHRA, EZBNMHRC and EZBNMRHP have changed and two request options have been added:

– GetTnMonitorGroups: obtain information about TN3270E monitor groups

– GetTnConnectionData: obtain information about TN3270E connection performance data.

Filters for GetTnConnectionData have been changed, including the following existing filters:

- Resource ID, Server resource ID, Local IP address, Local IP address prefix, Local port, Remote IP address, Remote IP address prefix and Remote port.

Additional filters are available for GetTnConnectionData, including the following:

- 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 Telnet server the record contains the associated reason code. The *IP Configuration Reference* document (Appendix C: SMF type 119 records) provides more details. Note that two SMF sections appear the hostname 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.

Attention: Existing SMF119 telnet SNA session termination records (subtype 21) are larger than in previous releases even if the connection is not being monitored. Two new triplet headers have been added.

Problem determination for Telnet server as a standalone task

For all commands referred to in this section, see:

- *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

The following additional resources can help in problem determination:

- *z/OS Communications Server: IP and SNA Codes*, SC31-8791
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *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

Important: When a VARY TCPIP,,OBEYFILE command is issued, TELNETPARMS and BEGINVTAM blocks are both required for each port started or modified.

The following items can be useful when performing problem determination for the Telnet server:

- Reviewing the definition statements within the profile
- Using OBEYFILE to turn selected debug options on and off
- Using VARY DEBUG OFF to turn off all debug options
- Using VARY ABENDTRAP to set abend traps
- Using the MSG07 statement in the TN3270 profile
- Using SMF records to capture TN3270 connection activity
- Using the CTRACE system component to trace TN3270 processes

Reviewing the definition statements within the profile

If multiple Telnet servers are used (for example, multiple TCP/IP stacks in a SD environment), ensure that each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail or the cross-domain session request will fail.

The default PPT entry sets the Telnet server to non-cancelable. As a non-cancelable application, the Telnet 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 will be 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
```

Using OBEYFILE to turn selected debug options on and off

TN3270-specific debug messages can be turned on or off 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 1-49. The parameters can involve subparameters, which we do not show in our examples. For details on 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 1-49 DEBUG statement format

```
DEBUG parameter1  parameter2
```

As shown in Table 1-2, 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. The table does not imply a one-to-one relationship.

Table 1-2 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: The PROFILE parameter can be specified in TELNETGLOBALS only. It has no affect 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. Once data is obtained, issue another VARY TCPIP,,OBEYFILE command for a TN3270 profile that omits the DEBUG statement or specifies DEBUF 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. Once 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 1-50. Assume the profile member name for turning the trace on is TELNDBON. (In order not to change anything else in the current profile, TELNB30A, TELNDBON is a copy of TELNB30A, with these three statements added).

Example 1-50 Preparing a DEBUG TRACE statement for OBEYFILE command

```
profile statements in member TELNDBON
   . . .
PARMSGROUP DEBUGIT DEBUG TRACE JOBLOG ENDPARMSGROUP
IPGROUP DEBUGIPGRP 10.20.1.241 ENDIPGROUP
PARMSMAP DEBUGIT IPGRP,DEBUGIPGRP
```

```
   . . .

VARY TCPIP,TN3270B,TELNET,OBEYFILE,TCPIPB.TCPPARMS(TELNDBON)
   EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 624
               TCPIPB.TCPPARMS(TELNDBON)
   EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 625
               TCPIPB.TCPPARMS(TELNDBON)
   EZZ6018I TELNET PROFILE UPDATE COMPLETE FOR PORT    23
   EZZ6038I TELNET 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 1-51.

Example 1-51   Display profile showing debug options ON

```
D TCPIP,TN3270B,T,CLID,PORT=23,PROFILE=ALL,DET,MAX=*
   EZZ6081I TELNET CLIENTID DISPLAY 689
   CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
   NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
   ------------------ ------ --------- -------- ---------- --------
   IPGRP
     DEBUGIPGRP
                           0 PARMSGRP  DEBUGIT             --------
   NULL
     NULL
                           0 DEFAPPL   TSO                 --------
   --------------------------------------- PROF: CURR CONNS:      0
   NULL
     NULL
                           1 DEFAPPL   TSO                 --------
   ----- PORT:    23  ACTIVE              PROF: 0001 CONNS:      1
   ------------------------------------------------------------
   12 OF 12 RECORDS DISPLAYED

D TCPIP,TN3270B,T,OBJ,PORT=23,PROFILE=ALL,DET,MAX=*
   D TCPIP,TN3270B,T,OBJ,PORT=23,PROFILE=ALL,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 691
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   ARAPPL
    SC30N*          0
                                                            -A------
    NVAS*           0
                                                            -A--Q---
    TSO*            0
                                                            -A-D----
    *               0
                                                            -A------
    *DEFAPPL        0
                                                            -A------
   DEFAPPL
    TSO             0 NULL      NULL
                                                            --------
   LUGRP
    *DEFLUS*        0
                                                            --------
   PARMSGRP
    DEBUGIT         0 IPGRP     DEBUGIPGRP
                                                            --------
    *DEFAULT         -------NO MAPPING---------
                                                    --------
    *TGLOBAL         -------NO MAPPING---------
                                                    --------
    *TPARMS          -------NO MAPPING---------
                                                    --------
   ------------------------------------- PROF: CURR CONNS:      0
```

The profile statements and obeyfile command for deactivating the trace is shown in Example 1-52. Assume the profile member name for turning the trace off is TELNDBOF.

Example 1-52 Preparing a DEBUG EXCEPTION statement for OBEYFILE command

```
profile statements in member TELNDBOF
   . . .
PARMSGROUP DEBUGIT DEBUG EXCEPTION ENDPARMSGROUP
IPGROUP DEBUGIPGRP 10.20.1.241 ENDIPGROUP
PARMSMAP DEBUGIT IPGRP,DEBUGIPGRP
   . . .

VARY TCPIP,TN3270B,TELNET,OBEYFILE,TCPIPB.TCPPARMS(TELNDBOF)
   EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 861
              TCPIPB.TCPPARMS(TELNDBOF)
   EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 862
              TCPIPB.TCPPARMS(TELNDBOF)
   EZZ6018I TELNET PROFILE UPDATE COMPLETE FOR PORT    23
   EZZ6038I TELNET COMMAND OBEYFILE  COMPLETE
```

In order not to change 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).

Using 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 due to 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 1-53.

Example 1-53 DEBUG OFF command used to turn off all debug options

```
V TCPIP,TN3270B,T,DEBUG,OFF
   EZZ6038I TELNET COMMAND DEBUG OFF COMPLETE
```

The options are listed again, after the DEBUF OFF command, as shown in Example 1-54.

Example 1-54 Display PROFILE to see DEBUG settings after DEBUG OFF command

```
D TCPIP,TN3270B,T,PROF,PORT=23,DET
     .........
   DIAGNOSTICS
      DEBUG            VARYOFF
      DEBUG ROUTING    CTRACE
      FULLDATATRACE
     .........
```

Using 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. Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details related to the use of this command.

Using the MSG07 statement in the TN3270 profile

The MSG07 parameter is very 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 end user without additional help. Even for more difficult problems, MSG07 provides a good starting point. We recommend that MSG07 always be coded in order 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 starting with CS for z/OS V1R8.

MSG07 being used in the TELNETPARMS block is shown in Example 1-55.

Example 1-55 MSG07 used in TELNETPARMS block

```
TELNETPARMS
   . . .
   MSG07
   . . .
ENDTELNETPARMS
```

For details on its use and syntax, see the MSG07 statement description in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Using SMF records to capture TN3270 connection activity

SMF records are written when an end user establishes a session (SMF LOGN or Telnet server SNA Session Initiation record, subtype 20) and when the session is ended (SMF LOGF or Telnet server 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 1-56. Example 1-56 turns off all 118 format records and uses standard 119 format records.

Example 1-56 SMFINIT/SMFTERM used in TELNETPARMS block

```
TELNETPARMS
   . . .
   SMFINIT 0 SMFINIT TYPE119
   SMFTERM 0 SMFTERM TYPE119
   . . .
ENDTELNETPARMS
```

For details on its use and syntax, see the SMFINIT/SMFTERM statement descriptions in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Using 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. Refer to the following sources for details:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342
- *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis

Using 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. Refer to the following sources for details:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342
- *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis

Using the CTRACE system component to trace TN3270 processes

If a problem is not resolved using the above tools, IBM service will likely need a CTRACE with option Telnet. CTRACE, with only the Telnet option, gives very complete information about the TN3270 processes. To debug almost any TN3270 problem no other CTRACE option is needed. Generally, the other options simply take up space, creating a trace-wrap condition more quickly, especially if TN3270 is running as part of the TCP/IP stack. 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, refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

Use *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis.

1.4.2 Implementing the Telnet server with connection security features

The Telnet server supports various methods and levels of connection security.

The following topics discuss Implementing the Telnet server with connection security features:

- Implementation tasks for the Telnet server with connection security
- Configuration examples for the Telnet server with connection security
- Verification for the Telnet server with connection security
- Problem determination for the Telnet server with connection security

Implementation tasks for the Telnet server with connection security

The following tasks are discussed:

- Data overrun security
- Transport Layer Security
- Network Access Control

Data overrun security

The MAXRECEIVE, MAXVTAMSENDQ, MAXREQSESS, and MAXRUCHAIN parameters can be coded at all three parameter block levels (TELNETGLOBALS, TELNETPARMS, and PARMSGROUP) for different levels of granularity. Refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776, for syntax details on each statement.

Transport Layer Security

The following tasks are required to enable TLS/SSL support for TN3270, with server authentication:

- Generate the Telnet server private key and server certificate by either of the following methods:
  - Using the GSKKYMAN utility to create a keyring file. If you choose to use GSKKYMAN then refer to:
    - *z/OS Communications Server: IP Configuration Guide*, SC31-8775
    - *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342
  - Using RACF utilities to create a RACF controlled certificate database. If you choose to use RACF then refer to:
    - *z/OS Communications Server: IP Configuration Guide*, SC31-8775
    - *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342
- Use the KEYRING statement to define the type and location of the key ring just created.
  - KEYRING HFS */usr/ssl/server.kdb*: In this example, two files, server.kdb and server.sth, were created using the gskkyman utility. The server's certificate is contained in the server.kdb file and designated as the default certificate. The key database and the password stash file must reside in the same directory.
  - KEYRING SAF *serverkeyring*: In this example, RACF is used to manage keys and certificates. The server certificate is connected to a key ring called SERVERKEYRING and designated as the default certificate.
  - The KEYRING statement can be coded in the TELNETGLOBALS or TELNETPARMS statement blocks.
  - The KEYRING statement is valid only with a secure port.
  - All uses of the KEYRING statement must specify the same data type and name. If coded in the TELNETGLOBALS statement block, any TELNETPARMS KEYRING values must match. If they do not, the port update is rejected.

   – If all secure ports (specified on the SECUREPORT statement) are in a *stopped* status when a profile update occurs, the KEYRING file is refreshed when a new secure port is activated.

   > **Note:** If a secure port is active during a profile update, the KEYRING name cannot change. If a change is attempted, an error message is issued for this parameter and the profile update for the related port is rejected. To change the KEYRING name, *all* secure ports must first be stopped.

- ConfigureTN3270 to include one or more TLS/SSL-enabled ports and specify the name of the key ring created in the step above in the TELNETGLOBALS block or the TELNETPARMS block.

  The Telnet server profile configuration must be updated with appropriate statements supporting the security features selected. There are two essential profile statements required for establishing secure ports:

  – SECUREPORT: All TLS/SSL-enabled TN3270 ports must be defined by specifying a TELNETPARMS block for each port. The SECUREPORT port designation statement in the TELNETPARMS block indicates the port is capable of handling SSL connections.

  – KEYRING: A server certificate is required for the server authentication process defined by the SSL protocol. This certificate is stored in a key ring. The key ring type and location is specified in the KEYRING statement. Only one key ring can be used by the Telnet server.

- Configure optional security parameters. Refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776, for details on statement syntax.

  > **Note:** Optional security parameters can only be specified for SECUREPORTs and can be specified in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP blocks. The parameters specified in the PARMSGROUP block apply only to the clients mapped to the PARMSGROUP block by the PARMSMAP statement and override the parameters specified in the TELNETPARMS or TELNETGLOBALS block. The parameters specified in the TELNETPARMS block apply to any connection for that port if not overridden by a PARMSGROUP parameter. The parameters specified in the TELNETGLOBALS block apply to any connection for any port if not overridden by a TELNETPARMS or PARMSGROUP parameter.

  Three important optional profile statements can be used to assist in defining the types and level of security imposed on the port connections:

  – CONNTYPE: The type of secure connection to be used on the port is specified with this statement. If it is specified in the TELNETPARMS block, it sets the default type for the port. If it is specified in a PARMSGROUP block, it can override the port's default setting. The types that can be specified are:

    • SECURE
    • NEGTSECURE
    • BASIC
    • ANY
    • NONE

  – CLIENTAUTH: The CLIENTAUTH parameter indicates that the client must send a client certificate to the server. If this parameter is not specified, a client certificate is not requested during the SSL handshake and no certificate-based client authentication is done. The level of validation performed depends on the option specified:

- SSLCERT
- SAFCERT
- NONE

– ENCRYPTION/ENDENCRYPTION: The Telnet server has an ordered list of encryption algorithms it will use by default, beginning with the lowest level of encryption to the highest, until it synchronizes (negotiates) to one that is compatible with the client. Use this statement to alter the order and to limit the list of supported algorithms. Perhaps you need to limit the choices to the two highest levels of encryption, not allowing the lower levels to even be attempted. If this parameter is not specified, all encryption algorithms that can be specified will be negotiated by TN3270, from low-level to high-level encryption. Each z/OS system level supports a specific set of encryption algorithms.The algorithms that can be specified are shown in Table 1-3.

Table 1-3 Encryption algorithms: default ordered list

ENCRYPTION algorithm_name	Abbreviation in Telnet displays
SSL_RC4_SHA	4S
SSL_RC4_MD5	4M
SSL_AES_256_SHA	A2
SSL_AES_128_SHA	A1
SSL_3DES_SHA	3S
SSL_DES_SHA	DS
SSL_RC4_MD5_EX	4E
SSL_RC2_MD5_EX	2E
SSL_NULL_SHA	NS
SSL_NULL_MD5	NM
SSL_NULL_Null	NN

– There are more statements available for customizing secure ports. For a complete list of parameter statements and their syntax, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For discussions on their use, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

- Restart the Telnet server task or issue the VARY TCPIP,,OBEYFILE command with the updated configuration file.

Network Access Control

When TN3270 is modified with a VARY TCPIP,,OBEYFILE command, the NACUSERIDs are reverified for the TN3270 ports defined in the data set referenced by the command. If a TN3270 port has NACUSERID NAC_name_1, you cannot use the VARY TCPIP,,OBEYFILE command to change that port's NACUSERID to NAC_name_2. The port must first be stopped, and then started with the new NAC_name_2 using the VARY TCPIP,,OBEYFILE command. On a restricted TCP/IP stack (not SYSMULTI), the NACUSERID will run under the SECLABEL of the TCP/IP stack.

On an unrestricted TCP/IP stack (SYSMULTI), the NACUSERID will run under the default SECLABEL defined in the user profile. If no default SECLABEL is configured in the user profile, SYSLOW is used by default. The NACUSERID user profile must be permitted to the SECLABEL it is to run under or port activation will fail.

The NETACCESS statement in the TCP/IP profile is used to configure portions of your IP network into named security zones. Each defined security zone must have a SERVAUTH profile for the resource named EZB.NETACCESS.*sysname.tcpname.zonename*. The user ID associated with the TN3270 port must have READ access to the security zone that maps its bind address (0.0.0.0/32 for INADDR_ANY unless overridden by the PORT statement in the TCP/IP profile) and to every security zone that maps client IP addresses that TN3270 is to accept connections from on this port.

Configuration examples for the Telnet server with connection security

The security-related TN3270 profile TELNETPARMS statements that had to be added for port 992 are shown in Example 1-57.

Example 1-57 TN3270 profile TELNETPARMS block added for connection security on port 992

```
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING HFS /etc/sc30.keyring.kdb       ;keyring used by all secure
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
ENDTELNETPARMS
```

The PARMSGROUP blocks that were added for port 992 are shown in Example 1-58. They are examples of defining objects that will be mapped later to client IDs. In this particular case, they are object groups being mapped to client ID groups. There are four object groups being defined:

- General user - Accesses port 992 on destination 10.20.10.21, requiring no SSL
- Admin - Accesses port 992 on destination 10.20.10.22, requiring plain SSL
- Payroll - Accesses port 992 on destination 10.20.10.23, requiring client authentication
- Shipping - Accesses port 992 on destination 10.20.1.230, decides at connection time

The destination addresses that belong to subnet 10.20.10.* are Dynamic VIPA addresses defined by the TCP/IP stack. They are not defined as Distributed Dynamic VIPAs in the stack. This stack owns them but does not attempt to distribute them in this scenario. And at this time no other stack defines them or provides backup for them.

The 10.20.1.230 address is the static VIPA address of the TCPIPB stack on SC30.

Example 1-58 TN3270 profile PARMSGROUP blocks added for connection security on port 992

```
; ---------------------------------------------------------------------
; This NOSSL        group is mapped to use no SSL security.           -
; LUSESSIONPEND = Force a requeue back to Defaultappl upon logoff      -
; ---------------------------------------------------------------------
  PARMSGROUP NOSSL
     LUSESSIONPEND
     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
```

The client ID groups are defined by identifying the clients that access the stack using the four destination IP addresses, as shown in Example 1-59. The are a number of alternative mapping methods to define client ID groups. This scenario uses the DESTIPGRP method.

Example 1-59 TN3270 profile security objects map to DESTIPGRPs accessing port 992

```
 DESTIPGROUP GENERALUSER 10.20.10.21   ENDDESTIPGROUP ; D-VIPA
  DESTIPGROUP ADMIN       10.20.10.22   ENDDESTIPGROUP ; D-VIPA
  DESTIPGROUP PAYROLL     10.20.10.23   ENDDESTIPGROUP ; D-VIPA
  DESTIPGROUP SHIPPING    10.20.1.230   ENDDESTIPGROUP ; Static VIPA
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP ; Any other 10. intface

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLP0  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER      DESTIPGRP,SHIPPING
  DEFAULTAPPL TSO       DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE
 ;-----------------------------------------------------------------
 ; 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.                    -
 ;-------------------------------------------------------------------
  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.
```

Any user who accesses the stack over any IP address not defined explicitly in this profile will be presented with the Network Solicitor panel and be prompted for a user ID, password, and desired application. No SSL security will be implemented for that type of connection.

The additional TCP/IP stack profile statements that had to be added are shown in Example 1-60. They were added in order for the stack to be able to own them, to advertise their existence to OSPF, and to accept connection requests from clients.

Example 1-60 TCP/IP profile VIPADYNAMIC statements added to support TN3270 SSL clients

```
VIPADYNAMIC
. . . . . . . . . .
  ;-------------------------------------------------------------------
  ;  Define addresses to be used with TN3270E Telnet server and SSL          -
  ;               (10.20.10.21  thru .23)                             -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payroll

 ENDVIPADYNAMIC
```

See Appendix D, “Configuration files: TN3270E Telnet server with connection security scenario” on page 351, for a complete listing of the started task procedures and profiles used for this scenario.

Verification for the Telnet server with connection security

Refer to “Verification steps for the Telnet server as a standalone task” on page 33.

Additional DISPLAY commands are used to show the security settings of a secure port. For a complete list of available TELNET-related commands and their syntax, see *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781.

Most of the display commands discussed below can be specified with either the SUMMARY or the DETAIL option. The summary option produces an abbreviated report showing settings without a descriptive legend. The detail option produces the complete descriptive legend. It is sometimes difficult to interpret the summary output because the option settings are represented by only one character and the legend is not included. To obtain the specific descriptive legend associated with your summary command, simply change the command to a detail command. Specify DETAIL instead of SUMMARY.

The following commands can be useful when validating secure port information in the Telnet server environment:

- Use TELNET CONN displays to show TN3270 connections.
- Display PROF to show profile SSL information.
- Display CLIENTID to show client group SSL information.
- Display OBJECT to show object SSL information.
- Display CONN to show connection SSL information.

Use TELNET CONN displays to show TN3270 connections

In preparation for the following displays, we have connected two clients to the TN3270B server on SC30, which is running under current TN3270 profile TELNB30B. We have defined Dynamic VIPA 10.20.10.21/24 to represent NetView® on SC30 (SC30N) with NOSSL support (BASIC). We have defined Static VIPA 10.20.1.230/30 to represent TSO on SC30 (SC30TS) with optional SSL support (UP2USER).

Using the TSO Telnet client on SC31, we connected to 10.20.10.21 port 922 (being mapped to NetView on SC30). Using the TSO Telnet client on SC32, we connected to 10.20.1.230, port 992 (being mapped to TSO on SC30). A display of the connections in Example 1-61 shows CONN=2063 from SC31 mapped to NetView on SC30, with an LU name of SC30B*S*08, the *S* indicating the LU pool for port 992 as expected. CONN=2067, from SC32, is mapped to TSO on SC30, with an LU name of SC30B*S*09, the *S* indicating the LU pool for port 992 as expected.

Because destination address 10.20.10.21 is associated with parmsgroup NOSSL, connection 2063 did not perform any SSL handshake, and thus no encryption type is indicated. Even though destination address 10.20.1.230 is associated with parmsgroup UP2USER (SSL optional), the TSO Telnet client does not negotiate or request SSL. Therefore, connection 2067 did not perform any SSL handshake, and thus no encryption type is indicated.

Connection 1FD1 is from a Personal Communications terminal configured to connect to destination IP address 10.20.10.22, port 992, and to request SSL without client authentication. Notice the encryption type is 4S for connid=1FD1. The SSLPLAIN parmsgroup is mapped to USS table USSTEST1. The connection shows `pending` (T*P*E) with no APPLID or LOGMODE assigned while the USSTEST1 MSG10 is displayed. Then the user selects an application (NVAS in our case), and TN3270 fills in the applid name and the associated logmode as shown in Example 1-73 on page 70 and Example 1-74 on page 71.

Example 1-61 SC30 connections, SSL port 992 summary

```
D,TCPIP,TN3270B,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 477
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   00002067    ::FFFF:10.20.5.226..1033
                                      SC30BS09 SC30TS08  TA3 SNX32704
   00002063    ::FFFF:10.20.1.241..1031
                                      SC30BS08 SC30N008  TA3 SNX32704
   00001FD1 4S ::FFFF:9.12.4.221..3235
                                      SC30BS07           TPE
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   8 OF 8 RECORDS DISPLAYED
```

Display PROF to show profile SSL information

The PROFILE display command enables you to determine what profile-wide options are in effect for each profile, which profiles are still being used, and how many users are on each profile. Look for SSL settings, as highlighted in the profile summary report in Example 1-62.

Example 1-62 Display Telnet PROFILE for SSL information, summary

```
D TCPIP,TN3270B,T,PROF,PORT=992,SUM,MAX=*
   EZZ6060I TELNET PROFILE DISPLAY 512
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     *M*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD*
   ----- PORT:   992  ACTIVE               PROF: CURR CONNS:      3
   ------------------------------------------------------------
      FORMAT            LONG
      TCPIPJOBNAME      TCPIPB
      TNSACONFIG        DISABLED
      KEYRING           HFS /etc/sc30b.keyring.kdb
   7 OF 7 RECORDS DISPLAYED
```

The profile detail report shows more security specifics as highlighted in Example 1-63.

Example 1-63 Display Telnet PROFILE for SSL information, detail

```
D TCPIP,TN3270B,T,PROF,PORT=992,DET,MAX=*
   DIAGNOSTICS
   . . . . . .
   SECURITY
     SECUREPORT            992
     CONNTYPE          SECURE
     KEYRING           HFS /etc/sc30b.keyring.kdb
     CRLLDAPSERVER     NONE
     ENCRYPTION        4S,4M,A2,A1,3S,DS,4E,2E,NS,NM,NN (DEF)
     CLIENTAUTH        NONE
     NOEXPRESSLOGON
     NONACUSERID
     NOSSLV2
   . . . . . .
   TIMERS
   . . . . . .
```

Display CLIENTID to show client group SSL information

The CLIENTID display can be used to see what client IDs are defined in the profile and details about the client ID, such as a DESTIPGRP. Look for SSL information, as highlighted in Example 1-64.

Example 1-64 Display Telnet CLIENTID for SSL information, detail

```
D TCPIP,TN3270B,T,CLID,PORT=992,DET,MAX=*
   EZZ6081I TELNET CLIENTID DISPLAY 540
   CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
   NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
   ------------------ ------ --------- -------- ---------- --------
   DESTIPGRP
     GENERALUSER
                          1 DEFAPPL   SC30N               --------
     GENERALUSER
                          1 PARMSGRP  NOSSL               --------
     ADMIN
                          1 USS       USSTEST1            --------
     ADMIN
                          1 PARMSGRP  SSLPLAIN            --------
     PAYROLL
                          0 DEFAPPL   CICSCLP0            --------
     PAYROLL
                          0 PARMSGRP  SSLCERTS            --------
     SHIPPING
                          1 DEFAPPL   TSO                 --------
     SHIPPING
                          1 PARMSGRP  UP2USER             --------
```

```
    ANY1ELSE
                      1 PARMSGRP  NOSSL              --------
  ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
  ------------------------------------------------------------
  21 OF 21 RECORDS DISPLAYED
```

A client ID summary report can show a specific client group and the names that make it up, as seen in Example 1-65.

Example 1-65 Display Telnet CLIENTID for SSL information, summary

```
D TCPIP,TN3270B,T,CLID,PORT=992,TYPE=DESTIPGRP,SUM,MAX=*
  EZZ6082I TELNET CLIENTID LIST 569
  DESTIPGRP
    GENERALUSER       ADMIN             PAYROLL
    SHIPPING          ANY1ELSE
  ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
  ------------------------------------------------------------
  5 OF 5 RECORDS DISPLAYED
```

Display OBJECT to show object SSL information

The OBJECT display can be used to see what objects are defined in the profile and some details about the object. If you specify a TYPE (which you know is related to a secure group you defined, such as DEFAPPL, PARMSGRP, or USS), you can see SSL-related information. Some examples are shown in Example 1-66.

Example 1-66 Display Telnet OBJECT for SSL information, summary

```
D TCPIP,TN3270B,T,OBJ,PORT=992,SUM,MAX=*
  EZZ6084I TELNET OBJECT LIST 586
  ARAPPL
    SC30N*    NVAS*     TSO*      *
  DEFAPPL
    SC30N     CICSCLP0  TSO
  PRTAPPL
    NO OBJECTS
  LINEAPPL
    NO OBJECTS
  MAPAPPL
    NO OBJECTS
  USS
    USSTEST1
  INT
    NO OBJECTS
  LU
    NO OBJECTS
  LUGRP
    *DEFLUS*
  APPLLUG
    NO OBJECTS
  PRT
    NO OBJECTS
  PRTGRP
    NO OBJECTS
  PARMSGRP
    NOSSL     SSLPLAIN  SSLCERTS  UP2USER   *DEFAULT  *TGLOBAL
    *TPARMS
  MONGRP
    NO OBJECTS
```

```
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   31 OF 31 RECORDS DISPLAYED
```

The object detail report can be filtered to show a specific object type and the client IDs mapped to the type, as seen in Example 1-67.

Example 1-67 Display Telnet OBJECT for SSL information, DEFAPPL

```
D TCPIP,TN3270B,T,OBJ,PORT=992,TYPE=DEFAPPL,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 613
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   DEFAPPL
    SC30N           1 DESTIPGRP GENERALUSER
                                                            --------
    CICSCLP0        0 DESTIPGRP PAYROLL
                                                            --------
    TSO             1 DESTIPGRP SHIPPING
                                                            --------
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   9 OF 9 RECORDS DISPLAYED
```

The object report can be used to check which PARMSGROUPs are mapped to which client IDs. The report indicates how many connections are associated with which group, as seen in Example 1-68.

Example 1-68 Display Telnet OBJECT for SSL information, PARMSGRP

```
D TCPIP,TN3270B,T,OBJ,PORT=992,TYPE=PARMSGRP,DET,MAX=*
   D TCPIP,TN3270B,T,OBJ,PORT=992,TYPE=PARMSGRP,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 679
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   PARMSGRP
    NOSSL           1 DESTIPGRP GENERALUSER
                                                            --------
    NOSSL           0 DESTIPGRP ANY1ELSE
                                                            --------
    SSLPLAIN        1 DESTIPGRP ADMIN
                                                            --------
    SSLCERTS        0 DESTIPGRP PAYROLL
                                                            --------
    UP2USER         1 DESTIPGRP SHIPPING
                                                            --------
    *DEFAULT          -------NO MAPPING---------
                                                            --------
    *TGLOBAL          -------NO MAPPING---------
                                                            --------
    *TPARMS           -------NO MAPPING---------
                                                            --------
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   19 OF 19 RECORDS DISPLAYED
```

The object report can be used to check which USS tables are mapped to which client IDs. The report indicates how many connections are associated with which table, as seen in Example 1-69.

Example 1-69 Display Telnet OBJECT for SSL information, USS

```
D TCPIP,TN3270B,T,OBJ,PORT=992,TYPE=USS,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 699
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   USS
    USSTEST1        1 DESTIPGRP ADMIN
                                                            --------
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
5 OF 5 RECORDS DISPLAYED
```

Display CONN to show connection SSL information

The CONNECTION display command without the CONN= parameter specified gives you a high-level view of what connections exist and what they are being used for. Look for EN/TY, Encryption Type. The connection command shows current connections and associated resources such as their LU name, Logmode, and application being used, as seen in Example 1-70.

Example 1-70 Display Telnet CONN for connection overview information

```
D TCPIP,TN3270B,T,CONN,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 719
            EN                                           TSP
   CONN     TY IPADDR..PORT           LUNAME   APPLID    PTR LOGMODE
   -------- -- --------------------- -------- --------  --- --------
   00002067    ::FFFF:10.20.5.226..1033
                                     SC30BS09 SC30TS08  TA3 SNX32704
   00002063    ::FFFF:10.20.1.241..1031
                                     SC30BS08 SC30N008  TA3 SNX32704
   00001FD1 4S ::FFFF:9.12.4.221..3235
                                     SC30BS07           TPE
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   8 OF 8 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and SUM specified gives you a summary look at one connection. It shows summary information regarding a single connection. Look for SSL information. An example is shown in Example 1-71.

Example 1-71 Display Telnet CONN SUMMARY for SSL information

```
D TCPIP,TN3270B,T,CONN,CONN=00001FD1,SUM,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 767
            EN                                           TSP
   CONN     TY IPADDR..PORT           LUNAME   APPLID    PTR LOGMODE
   -------- -- --------------------- -------- --------  --- --------
   00001FD1 4S ::FFFF:9.12.4.221..3235
                                     SC30BS07           TPE
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and DET specified gives you a complete look at one connection. It shows all the information available regarding a single connection. Look for SSL information. An example is shown in Example 1-72.

Example 1-72 Display Telnet CONN DETAIL for SSL information, before USS selection is made

```
D TCPIP,TN3270B,T,CONN,CONN=00001FD1,DET,MAX=*
   EZZ6065I TELNET CONNECTION DISPLAY 783
     CONNECTED: 12:41:40  08/21/2006  STATUS: SESSION PENDING
     CLIENT IDENTIFIER FOR CONN: 00001FD1   SECLABEL: **N/A**
       CLIENTAUTH USERID: **N/A**
       HOSTNAME: NO HOSTNAME
       CLNTIP..PORT: ::FFFF:9.12.4.221..3235
       DESTIP..PORT: ::FFFF:10.20.10.22..992
       LINKNAME: VIPL0A140A16
     PORT:   992 QUAL: NONE
       AFFINITY: TCPIPB
       STATUS: ACTIVE  SECURE        ACCESS: SECURE  4S SSLV3
     PROTOCOL: TN3270E          DEVICETYPE: IBM-3278-2-E
       TYPE: TERMINAL GENERIC
       OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                          NEWENV FUNCTIONS: --
     LUNAME: SC30BS07
     APPL: **N/A**
       USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
       LOGMODES  TN REQUESTED:           APPL SPECIFIED:
     MAPPING TYPE:  CONN IDENTIFIER
                         OBJECT   ITEM SPECIFIC     OPTIONS
                   **N/A**
                        >*DEFLUS*                   --------
       DEFLT APPL: **N/A**
       USS TABLE:  DG ADMIN
                         USSTEST1                   --------
       INT TABLE:  **N/A**
       PARMS:
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     *******  **TSBTQ***T  EC*  BB**D****  ***STS  *DD* *DEFAULT
     -------  ----------T  ---  ---------  ------  ---- *TGLOBAL
     -M-----  ---S-------  --F  SSH------  *--ST-  S--- *TPARMS
     *M*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* TP-CURR
       PARMSGROUP: DG ADMIN
     L------  -----------  ---  -S-------  ------  ---- SSLPLAIN
     LM*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* <-FINAL
   37 OF 37 RECORDS DISPLAYED
```

Once the user selects an application, TN3270 fills in the applid name and associated logmode, as shown in Example 1-73 and Example 1-74 on page 71.

Example 1-73 Connection summary information for SSL port 992 after USSMSG10 appl is selected

```
D TCPIP,TN3270B,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 894
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   00002067    ::FFFF:10.20.5.226..1033
                                     SC30BS09 SC30TS08  TA3 SNX32704
   00002063    ::FFFF:10.20.1.241..1031
```

```
                                   SC30BS08 SC30N008  TA3 SNX32704
   00001FD1 4S ::FFFF:9.12.4.221..3235
                                   SC30BS07 SC38EMS   TAE SNX32702
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   8 OF 8 RECORDS DISPLAYED
```

Example 1-74 Connection detail information for SSL port 992 after USSMSG10 appl is selected

```
D TCPIP,TN3270B,T,CONN,CONN=00001FD1,DET
   EZZ6065I TELNET CONNECTION DISPLAY 195
     CONNECTED: 12:41:40  08/21/2006  STATUS: SESSION ACTIVE
     CLIENT IDENTIFIER FOR CONN: 00001FD1   SECLABEL: **N/A**
       CLIENTAUTH USERID: **N/A**
       HOSTNAME: NO HOSTNAME
       CLNTIP..PORT: ::FFFF:9.12.4.221..3235
       DESTIP..PORT: ::FFFF:10.20.10.22..992
       LINKNAME: VIPL0A140A16
     PORT:   992 QUAL: NONE
       AFFINITY: TCPIPB
       STATUS: ACTIVE  SECURE        ACCESS: SECURE  4S SSLV3
     PROTOCOL: TN3270E          DEVICETYPE: IBM-3278-2-E
       TYPE: TERMINAL GENERIC
       OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                          NEWENV FUNCTIONS: --
     LUNAME: SC30BS07
     APPL: SC38EMS
       USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
       LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
     MAPPING TYPE:  CONN IDENTIFIER
                         OBJECT   ITEM SPECIFIC     OPTIONS
                   **N/A**
                         >*DEFLUS*                  --------
       DEFLT APPL: **N/A**
       USS TABLE:  DG ADMIN
                         USSTEST1                   --------
       INT TABLE:  **N/A**
       PARMS:
     PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
    (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
     -------  -----------  ---  ---------  ------  ----
     *******  **TSBTQ***T  EC*  BB**D****  ***STS  *DD* *DEFAULT
     -------  ----------T  ---  ---------  ------  ---- *TGLOBAL
     -M-----  ---S-------  --F  SSH------  *--ST-  S--- *TPARMS
     *M*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* TP-CURR
       PARMSGROUP: DG ADMIN
     L------  -----------  ---  -S-------  ------  ---- SSLPLAIN
     LM*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* <-FINAL
   37 OF 37 RECORDS DISPLAYED
```

Problem determination for the Telnet server with connection security

Refer to “Problem determination for Telnet server as a standalone task” on page 52.

Refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

If multiple Telnet servers are used (for example, multiple TCP/IP stacks in a SD environment), ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail or the cross-domain session request will fail.

1.4.3 Implementing multiple Telnet servers with SD

Multiple Telnet servers can accept connections distributed from SD.

The following topics discuss implementing multiple Telnet servers with SD:

- Implementation tasks for Telnet servers with SD
- Configuration examples for Telnet servers with SD
- Verification for Telnet servers with SD
- Problem determination for Telnet servers with SD

Implementation tasks for Telnet servers with SD

Because each of the Telnet servers in this scenario runs as a standalone task, all of the implementation tasks for a standalone Telnet server apply. Refer to 1.4.1, “Implementing the Telnet server executing as a standalone task” on page 25 to prepare the basic Telnet server.

The additional tasks to implement SD for the Telnet servers follow:

- Customize a second TCP/IP stack started task procedure.
- Customize a second TCP/IP stack configuration profile data set.
- Customize a second Telnet server started task procedure.
- Customize a second Telnet server configuration profile data set.
- 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.
- Start the second (backup) stack on the second system.
- Start the second Telnet server on the second system.

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 proc name and same proclib for both stacks, and used system symbolics to provide unique names for the stacks’ configuration profile data sets.

See Appendix E, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 365, 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 SD backup stack. If you do not already have a second stack running, you must create it. SD support for our Telnet 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 in order to give them their uniqueness. Home IP addresses and possibly interface definitions are common differences. For a complete discussion and examples of setting up a stack, refer to:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

See Appendix E, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 365, for the basic stack profile used for this scenario.

Customize a second Telnet server started task procedure

This second started task is for running our second Telnet server. It can be modeled after our first server started task. We used the same proc name and same proclib for both servers, and used system symbolics to provide unique names for the stacks' configuration profile data sets.

See Appendix E, "Configuration files: Multiple TN3270E Telnet servers and SD scenario" on page 365, for the started task procedures used for this scenario.

Customize a second Telnet server configuration profile data set

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: TN3270 LU names to be used with VTAM must be unique for each server.

For a complete discussion and examples of setting up a Telnet server, refer to 1.4.1, "Implementing the Telnet server executing as a standalone task" on page 25.

See Appendix E, "Configuration files: Multiple TN3270E Telnet servers and SD scenario" on page 365, for the basic Telnet server profiles used for this scenario.

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. We used the same proc name and same proclib for both servers, and used system symbolics to provide unique names for the OMPROUTE configuration data sets.

See Appendix E, "Configuration files: Multiple TN3270E Telnet servers and SD scenario" on page 365, for the started task procedures used for this scenario.

Customize an OMPROUTE configuration to support 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 in order to give them their uniqueness: interface IP addresses and router IDs are examples. For details on configuring OMPROUTE, see:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *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. To review the statements added, see Example 1-79 on page 76 and Example 1-80 on page 77.

See Appendix E, "Configuration files: Multiple TN3270E Telnet servers and SD scenario" on page 365, for the basic OMPROUTE 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 10.20.20.0/24
```

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 D-VIPA, one will have to be allocated.

```
Our D-VIPA subnet is 10.20.10.0/24
```

Enable the primary stack to support sysplex functions

Add statements to the primary stack that enable SD support, as shown in Example 1-75.

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

The VIPADYNAMIC statements enable the stack to distribute Telnet server connections to the two servers. The statements we added are shown in Example 1-76.

Enable the backup stack to support sysplex functions

Add statements to the backup stack that enable SD support, as shown in Example 1-77 on page 75. This stack is the backup.

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 we added are shown in Example 1-78 on page 75.

Start the second (backup) stack on the second system

Issue the MVS START command on the second system:

```
S TCPIPB
```

Start the second Telnet server on the second system

Issue the MVS START command on the second system:

```
S TN3270B
```

Configuration examples for Telnet servers with SD

Example statements added to the two stacks and to the two Telnet server configuration data sets are shown below. Primary stack Sysplex statements are in Example 1-75.

Example 1-75 Sysplex enablement for primary stack

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.100 255.255.255.0 8
```

Primary stack VIPADYNAMIC statements are in Example 1-76.

Example 1-76 VIPADYNAMIC statements for primary stack

```
VIPADYNAMIC

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.20.10.1    PORT 20 21
                  DESTIP 10.20.20.100
```

```
                          10.20.20.101

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                          10.20.10.21   PORT 992
                  DESTIP 10.20.20.100
                          10.20.20.101

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                          10.20.10.22   PORT 992
                  DESTIP 10.20.20.100
                          10.20.20.101

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payrol
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                          10.20.10.23   PORT 992
                  DESTIP 10.20.20.100
                          10.20.20.101
  ;-------------------------------------------------------------------
  ;      Distribute to 10.20.20.100 via normal XCF   (no viparoute)  -
  ;      Distribute to 10.20.20.101 via IP routing   (   viparoute)  -
  ;-------------------------------------------------------------------
 ;;VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
   VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa

 ENDVIPADYNAMIC
```

Backup stack sysplex statements are shown in Example 1-77.

Example 1-77 Sysplex enablement for backup stack

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.101 255.255.255.0 8
```

Backup stack VIPADYNAMIC statements are in Example 1-78.

Example 1-78 VIPADYNAMIC statements for backup stack

```
VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                          10.20.10.1    PORT 20 21
                  DESTIP 10.20.20.100
```

```
                          10.20.20.101

  ;------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
  ;------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                          10.20.10.21   PORT 992
                   DESTIP 10.20.20.100
                          10.20.20.101

  ;------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
  ;------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                          10.20.10.22   PORT 992
                   DESTIP 10.20.20.100
                          10.20.20.101

  ;------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
  ;------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payrol
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                          10.20.10.23   PORT 992
                   DESTIP 10.20.20.100
                          10.20.20.101

  ;------------------------------------------------------------------
  ;      Distribute to 10.20.20.100 via IP routing   (   viparoute)  -
  ;      Distribute to 10.20.20.101 via normal XCF   (no viparoute)  -
  ;------------------------------------------------------------------
   VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa

 ENDVIPADYNAMIC
```

OMPROUTE statements added for D-XCF and D-VIPA support are shown below. Example 1-79 shows those for the primary stack.

Example 1-79 OMPROUTE additional statements to support D-XCF and D-VIPA: 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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
; ************************************************************
; *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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* 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=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
```

Example 1-80 shows those for the backup stack.

Example 1-80 OMPROUTE additional statements to support D-XCF and D-VIPA: 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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
; ************************************************************
; *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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* 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=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
```

Verification for Telnet servers with SD

Because each of the Telnet servers in this scenario runs as a standalone task, all of the verification tasks for a standalone Telnet server apply. Refer to “Verification steps for the Telnet server as a standalone task” on page 33.

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 on the NETSTAT command, refer to *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781.

The following additional tasks to verify sysplex distribution to Telnet servers are discussed below:

- 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.

Use TELNET CONN displays to show TN3270 connections

In preparation for the following displays, we have connected two clients to the Dynamic VIPA 10.20.10.21 being distributed by SC30, which is running with stack profile PROFB30C and TN3270 profile TELNB30C. The D-VIPA 10.20.10.21 represents NetView on SC30 (SC30N). The distribution method for 10.20.10.21 port 992 is set to ROUNDROBIN for our test in order to show the result of distribution upon our two client connections. Using the TSO Telnet client on SC31, we connected to 10.20.10.21 port 922 (expecting to be mapped to NetView on SC30). Using the TSO Telnet client on SC32, we again connected to 10.20.10.21 port 992 (mapping to NetView 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.

A display of the connections on SC30 in Example 1-81 on page 79 and Example 1-83 on page 79 show CONN=91 from SC31 (10.20.1.241) mapped to NetView on SC30, with an LU name of SC30B*S*02, the *S* indicating the LU pool for secure port 992 as expected.

A display of the connections on SC31 in Example 1-82 on page 79 and Example 1-84 on page 80 show CONN=8E from SC32 (10.20.2.222) mapped to NetView on SC30, with an LU name of SC31BS01, the *S* indicating the LU pool for secure port 992 as expected. The connection summary report for SC30 is shown in Example 1-81.

Example 1-81 SC30 connection to Dynamic VIPA 10.20.10.21, summary

```
RO SC30,D,TCPIP,TN3270B,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 236
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   00000091    ::FFFF:10.20.1.241..1029
                                      SC30BS02 SC30N009  TA3 SNX32704
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The connection summary report for SC31 is shown in Example 1-82.

Example 1-82 SC31 connection to Dynamic VIPA 10.20.10.21, summary

```
RO SC31,D,TCPIP,TN3270B,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 660
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   0000008E    ::FFFF:10.20.2.222..1026
                                      SC31BS01 SC30N008  TA3 SNX32704
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

Detailed information about one single connection on SC30 is shown in Example 1-83.

Example 1-83 SC30 connection to Dynamic VIPA with detailed information

```
RO SC30,D,TCPIP,TN3270B,T,CONN,CONN=91,DET

EZZ6065I TELNET CONNECTION DISPLAY 252
  CONNECTED: 00:52:12  08/21/2006 STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 00000091   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.20.1.241..1029
    DESTIP..PORT: ::FFFF:10.20.10.21..992
    LINKNAME: VIPL0A140A15
  PORT:   992 QUAL: NONE
    AFFINITY: TCPIPB
    STATUS: ACTIVE  SECURE        ACCESS: NON-SECURE
  PROTOCOL: TN3270           DEVICETYPE: IBM-3278-4-E
    TYPE: TERMINAL GENERIC
    OPTIONS: -TET----   3270E FUNCTIONS: *N/A*
                       NEWENV FUNCTIONS: --
  LUNAME: SC30BS02
  APPL: SC30N009
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32704  APPL SPECIFIED: SNX32704
  MAPPING TYPE:  CONN IDENTIFIER
                      OBJECT   ITEM SPECIFIC     OPTIONS
                **N/A**
                     >*DEFLUS*                   --------
    DEFLT APPL: DG GENERALUSER
                      SC30N                      --------
    USS TABLE:  **N/A**
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
 (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
  -------  -----------  ---  ---------  ------  ----
  *******  **TSBTQ***T  EC*  BB**D****  ***STS  *DD* *DEFAULT
  -------  ----------T  ---  ---------  ------  ---- *TGLOBAL
  -M-----  ---S-------  --F  SSH------  *--ST-  S--- *TPARMS
  *M*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* TP-CURR
    PARMSGROUP: DG GENERALUSER
  *------  -----------  ---  -B-------  ------  ---- NOSSL
  *M*****  **TSBTQ***T  ECF  SBH*D****  ***STS  SDD* <-FINAL
37 OF 37 RECORDS DISPLAYED
```

Detailed information about one single connection on SC31 is shown in Example 1-84.

Example 1-84 SC31 connection to Dynamic VIPA with detailed information

```
RO SC31,D,TCPIP,TN3270B,T,CONN,CONN=8E,DET

EZZ6065I TELNET CONNECTION DISPLAY 662
  CONNECTED: 00:51:50  08/21/2006  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 0000008E   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.20.2.222..1026
    DESTIP..PORT: ::FFFF:10.20.10.21..992
    LINKNAME: VIPL0A140A15
  PORT:   992 QUAL: NONE
    AFFINITY: TCPIPB
    STATUS: ACTIVE  SECURE        ACCESS: NON-SECURE
  PROTOCOL: TN3270            DEVICETYPE: IBM-3278-4-E
    TYPE: TERMINAL GENERIC
    OPTIONS: -TET----   3270E FUNCTIONS: *N/A*
                       NEWENV FUNCTIONS: --
  LUNAME: SC31BS01
  APPL: SC30N008
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32704  APPL SPECIFIED: SNX32704
  MAPPING TYPE:  CONN IDENTIFIER
                      OBJECT   ITEM SPECIFIC     OPTIONS
                **N/A**
                     >*DEFLUS*                   --------
    DEFLT APPL: DG GENERALUSER
                      SC30N                      --------
    USS TABLE:  **N/A**
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION     DIA  SECURITY  TIMERS  MISC
 (LMTGQAK)(OATSKTQSWHT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
  -------  -----------  ---  ---------  ------  ----
  *******  **TSBTQ***T  EC*  BB**D****  ***STS  *DD* *DEFAULT
  -------  ----------T  ---  ---------  ------  ---- *TGLOBAL
  -M-----  ---S-------  --F  SSH------  *--ST-  S--- *TPARMS
  *M*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* TP-CURR
    PARMSGROUP: DG GENERALUSER
  *------  -----------  ---  -B-------  ------  ---- NOSSL
  *M*****  **TSBTQ***T  ECF  SBH*D****  ***STS  SDD* <-FINAL
37 OF 37 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 1-85 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 1-85 NETSTAT VCRT on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VCRT
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.20.10.21..992
     SOURCE:  10.20.1.241..1029
     DESTXCF: 10.20.20.100
   DEST:      10.20.10.21..992
     SOURCE:  10.20.2.222..1026
     DESTXCF: 10.20.20.101
   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 seen in Example 1-86.

Example 1-86 NETSTAT VCRT on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VCRT
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.20.10.21..992
     SOURCE:  10.20.2.222..1026
     DESTXCF: 10.20.20.101
   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 1-87 on page 82 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: Notice that 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 1-87 NETSTAT VDPT on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VDPT
   DYNAMIC VIPA DESTINATION PORT TABLE:
   DEST:        10.20.10.1..20
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.1..20
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.1..21
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.1..21
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.21..992
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000002  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.20.10.21..992
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000001  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.20.10.22..992
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.22..992
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.23..992
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
     FLG: SERVERWLM
   DEST:        10.20.10.23..992
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
     FLG: SERVERWLM
   10 OF 10 RECORDS DISPLAYED
   END OF THE REPORT
```

SC31 is not a distributor at the moment, therefore it shows no information, as seen in Example 1-88.

Example 1-88 NETSTAT VDPT on system SC31

```
RO 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 1-89.

Example 1-89 NETSTAT VIPADCFG on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADCFG
   DYNAMIC VIPA INFORMATION:
     VIPA DEFINE:
       IPADDR/PREFIXLEN: 10.20.10.1/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.20.10.21/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.20.10.22/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.20.10.23/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
     VIPA RANGE:
       IPADDR/PREFIXLEN: 10.20.10.240/28
         MOVEABLE: NONDISR
     VIPA DISTRIBUTE:
       DEST:         10.20.10.1..20
         DESTXCF:    10.20.20.100
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.1..20
         DESTXCF:    10.20.20.101
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.1..21
         DESTXCF:    10.20.20.100
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.1..21
         DESTXCF:    10.20.20.101
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.21..992
         DESTXCF:    10.20.20.100
           SYSPT:    NO   TIMAFF: NO    FLG: ROUNDROBIN
       DEST:         10.20.10.21..992
         DESTXCF:    10.20.20.101
           SYSPT:    NO   TIMAFF: NO    FLG: ROUNDROBIN
       DEST:         10.20.10.22..992
         DESTXCF:    10.20.20.100
           SYSPT:    NO   TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.22..992
         DESTXCF:    10.20.20.101
           SYSPT:    NO   TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.23..992
         DESTXCF:    10.20.20.100
           SYSPT:    NO   TIMAFF: NO    FLG: SERVERWLM
       DEST:         10.20.10.23..992
         DESTXCF:    10.20.20.101
           SYSPT:    NO   TIMAFF: NO    FLG: SERVERWLM
     VIPA ROUTE:
       DESTXCF:      10.20.20.101
         TARGETIP:   10.20.1.241
```

```
END OF THE REPORT
```

The backup stack shows VIPA BACKUP, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 1-90.

Example 1-90 NETSTAT VIPADCFG on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADCFG
   DYNAMIC VIPA INFORMATION:
     VIPA BACKUP:
       IPADDR/PREFIXLEN: 10.20.10.1
         RANK: 200  MOVEABLE:            SRVMGR:
       IPADDR/PREFIXLEN: 10.20.10.21
         RANK: 200  MOVEABLE:            SRVMGR:
       IPADDR/PREFIXLEN: 10.20.10.22
         RANK: 200  MOVEABLE:            SRVMGR:
       IPADDR/PREFIXLEN: 10.20.10.23
         RANK: 200  MOVEABLE:            SRVMGR:
     VIPA RANGE:
       IPADDR/PREFIXLEN: 10.20.10.240/28
         MOVEABLE: NONDISR
     VIPA DISTRIBUTE:
       DEST:         10.20.10.1..20
         DESTXCF:    10.20.20.100
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.1..20
         DESTXCF:    10.20.20.101
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.1..21
         DESTXCF:    10.20.20.100
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.1..21
         DESTXCF:    10.20.20.101
           SYSPT:    YES  TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.21..992
         DESTXCF:    10.20.20.100
           SYSPT:    NO   TIMAFF: NO    FLG: ROUNDROBIN
       DEST:         10.20.10.21..992
         DESTXCF:    10.20.20.101
           SYSPT:    NO   TIMAFF: NO    FLG: ROUNDROBIN
       DEST:         10.20.10.22..992
         DESTXCF:    10.20.20.100
           SYSPT:    NO   TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.22..992
         DESTXCF:    10.20.20.101
           SYSPT:    NO   TIMAFF: NO    FLG: BASEWLM
       DEST:         10.20.10.23..992
         DESTXCF:    10.20.20.100
           SYSPT:    NO   TIMAFF: NO    FLG: SERVERWLM
       DEST:         10.20.10.23..992
         DESTXCF:    10.20.20.101
           SYSPT:    NO   TIMAFF: NO    FLG: SERVERWLM
     VIPA ROUTE:
       DESTXCF:      10.20.20.100
         TARGETIP:   10.20.1.230
   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

Examples of VIPADYN showing Dynamic VIPA and VIPAROUTE information follow.

Example 1-91 shows SC30, Netstat VIPADYN: ftp and TN3270 DVIPA addresses.

Example 1-91 NETSTAT VIPADYN on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.23/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
   VIPA ROUTE:
     DESTXCF: 10.20.20.101
       TARGETIP: 10.20.1.241
       RTSTATUS: ACTIVE
   5 OF 5 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 1-92 shows SC30, Netstat VIPADYN,DVIPA: filters on DVIPA only.

Example 1-92 NETSTAT VIPADYN,DVIPA on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.23/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
   4 OF 4 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 1-93 shows SC30, Netstat VIPADYN,VIPAROUTE: filters on viparoute only.

Example 1-93 NETSTAT VIPADYN,VIPAROUTE on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
   VIPA ROUTE:
     DESTXCF: 10.20.20.101
       TARGETIP: 10.20.1.241
       RTSTATUS: ACTIVE
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 1-94 shows SC31, Netstat VIPADYN: ftp and TN3270 DVIPA addresses.

Example 1-94 NETSTAT VIPADYN on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.23/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
   VIPA ROUTE:
     DESTXCF: 10.20.20.100
       TARGETIP: 10.20.1.230
       RTSTATUS: ACTIVE
   5 OF 5 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 1-95 shows SC31, Netstat VIPADYN,DVIPA: filters on DVIPA only.

Example 1-95 NETSTAT VIPADYN,DVIPA on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.23/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
   4 OF 4 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 1-96 shows SC31, Netstat VIPADYN,VIPAROUTE: filters on viparoute only.

Example 1-96 NETSTAT VIPADYN,VIPAROUTE on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
   VIPA ROUTE:
     DESTXCF: 10.20.20.100
       TARGETIP: 10.20.1.230
       RTSTATUS: ACTIVE
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

Problem determination for Telnet servers with SD

Refer to “Problem determination for Telnet server as a standalone task” on page 52.

Refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

If multiple Telnet servers are used (for example, multiple TCP/IP stacks in a SD environment), ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail, or the cross-domain session request will fail.

1.4.4 Implementing SD with GR applications

Multiple Telnet servers can receive distributed connections from SD. They can in turn send the connected sessions to an application that is an active member of a VTAM generic resource group.

The following topics discuss multiple Telnet servers and implementing sysplex distribution with GR Applications:

- Implementation tasks for multiple Telnet servers, SD, and GR
- Configuration examples for multiple Telnet servers, SD, and GR
- Verification for multiple Telnet servers, SD, and GR
- Problem determination for multiple Telnet servers, SD, and GR

Implementation tasks for multiple Telnet servers, SD, and GR

Because each of the Telnet servers in this scenario runs as a standalone task, all of the implementation tasks for a standalone Telnet server apply. Refer to 1.4.1, "Implementing the Telnet server executing as a standalone task" on page 25.

The implementation tasks for multiple Telnet servers with sysplex distribution apply as well. Refer to 1.4.3, "Implementing multiple Telnet servers with SD" on page 72.

There are no additional tasks to implement sysplex distribution to Telnet servers that point to applications participating in generic resources. The additional effort is on the application side where generic resource structures have to be defined and prepared for use in the sysplex.

The VTAM start option, STRGR, must point to the generic resource structure name. Refer to these additional resources:

- *z/OS Communications Server: SNA Operation*, SC31-8779
- *SNA in a Parallel Sysplex Environment,* SG24-2113

Configuration examples for multiple Telnet servers, SD, and GR

Example statements added to the two stacks and to the two Telnet server configuration data sets are shown in Example 1-97 through Example 1-102 on page 90. The standalone task scenario was the base for this scenario. These statements were added to the base standalone task profiles to achieve this scenario.

For a complete listing of the started task procedures and profiles used for this scenario, see Appendix E, "Configuration files: Multiple TN3270E Telnet servers and SD scenario" on page 365. The same files used for that scenario were used for this one as well. The following examples show only the changes made to the standalone scenario to create the environment for this scenario.

Example 1-97 shows primary stack sysplex enablement.

Example 1-97 Sysplex enablement for primary stack

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.100 255.255.255.0 8
```

Example 1-98 on page 88 shows primary stack Dynamic VIPA additions.

Example 1-98   VIPADYNAMIC statements for primary stack

```
VIPADYNAMIC

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-----------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.20.10.1    PORT 20 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
  ;-----------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.20.10.21   PORT 992
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
  ;-----------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.20.10.22   PORT 992
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
  ;-----------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payrol
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.20.10.23   PORT 992
                  DESTIP 10.20.20.100
                         10.20.20.101
  ;-----------------------------------------------------------------
  ;      Distribute to 10.20.20.100 via normal XCF   (no viparoute)  -
  ;      Distribute to 10.20.20.101 via IP routing   (   viparoute)  -
  ;-----------------------------------------------------------------
 ;;VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
   VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa

 ENDVIPADYNAMIC
```

Example 1-99 shows backup stack sysplex enablement.

Example 1-99   Sysplex enablement for backup stack

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.101 255.255.255.0 8
```

Example 1-100 on page 89 shows backup stack Dynamic VIPA additions.

Example 1-100   VIPADYNAMIC statements for backup stack

```
VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   **VIPABACKUP 200** MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.20.10.1    PORT 20 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   **VIPABACKUP 200** MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.20.10.21   PORT 992
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
  ;-------------------------------------------------------------------
   **VIPABACKUP 200** MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.20.10.22   PORT 992
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
  ;-------------------------------------------------------------------
   **VIPABACKUP 200** MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payrol
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.20.10.23   PORT 992
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-------------------------------------------------------------------
  ;      Distribute to 10.20.20.100 via IP routing   (   viparoute)  -
  ;      Distribute to 10.20.20.101 via normal XCF   (no viparoute)  -
  ;-------------------------------------------------------------------
   VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa
ENDVIPADYNAMIC
```

Example 1-101 shows OMPROUTE additions for the primary stack for D-XCF and D-VIPA.

Example 1-101   OMPROUTE additional statements to support D-XCF and D-VIPA: 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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ***********************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
; ***********************************************************
; *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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* 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=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
```

Example 1-102 shows OMPROUTE additions for the backup stack for D-XCF and D-VIPA.

Example 1-102 OMPROUTE additional statements to support D-XCF and D-VIPA: 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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ***********************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
; *************************************************************
; *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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* 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=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
```

See Appendix E, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 365, for a complete listing of the started task procedures and profiles used for this scenario.

Verification for multiple Telnet servers, SD, and GR

Because each of the Telnet servers in this scenario runs as a standalone task, all of the verification tasks for a standalone Telnet server apply. Refer to “Verification steps for the Telnet server as a standalone task” on page 33.

The verification tasks for multiple Telnet servers with sysplex distribution apply as well. Refer to “Verification for Telnet servers with SD” on page 78.

The additional tasks to verify sysplex distribution to Telnet servers that point to applications participating in generic resources are discussed here. Because the generic resources function is implemented by VTAM, the commands used to display GR information are VTAM display commands and application-specific commands.

The following methods are discussed:

- Use Display VTAMOPTS for STRGR to verify GR is active.
- Use Display RSCLIST,TYPE=GENERIC to list all GR resources.
- Use Display ID=grname to show GR group members and their statuses.
- Use Display GRAFFIN to show GR affinities.
- Use Modify GR to remove an application from the GR structure.

Use Display VTAMOPTS for STRGR to verify GR is active

Display the setting of the VTAM start option, STRGR, to determine if generic resources support is active, as shown in Example 1-103.

Example 1-103 Display VTAMOPTS, OPT=STRGR to show the setting

```
D NET,VTAMOPTS,OPT=STRGR
   IST097I DISPLAY ACCEPTED
   IST1348I VTAM STARTED AS INTERCHANGE NODE
   IST1189I STRGR    = ISTGENERIC
IST314I END
```

Use Display RSCLIST,TYPE=GENERIC to list all GR resources

Display all generic resources known locally (type GENERIC USERVAR) and in the sysplex (type GENERIC RESOURCE), as shown in Example 1-104.

Example 1-104 Display RSCLIST,TYPE=GENERIC shows known generic resources group names

```
D NET,RSCLIST,ID=*,IDTYPE=GENERIC
   IST097I DISPLAY ACCEPTED
   IST350I DISPLAY TYPE = RSCLIST
   IST1417I NETID     NAME     STATUS     TYPE              MAJNODE
   IST1418I USIBMSC   CICSCLP0 ACT/S      GENERIC RESOURCE  **NA**
   IST1418I USIBMSC   CICSCLU0 ACT/S      GENERIC RESOURCE  **NA**
   IST1418I USIBMSC   CICSLUT0 ACT/S      GENERIC RESOURCE  **NA**
   IST1418I USIBMSC   CICSLUT1 ACT/S      GENERIC RESOURCE  **NA**
   IST1454I 4 RESOURCE(S) DISPLAYED FOR ID=*
IST314I END
```

Use Display ID=grname to show GR group members and their statuses

The Display ID=*grname* command shows all known members of the GR group:

- Member name
- Owning CP name
- Whether the resource is currently available to be selected during resolution:
  - NO indicates that the generic resource is on an end node that does not have a CP-CP session with its network node server, and is therefore not selectable.
  - YES indicates the resource is selectable.
  - DEL indicates that the resource has deleted itself as a generic resource and is not selectable. If you need to fully delete the generic resource from VTAM and the generic resource coupling facility structure, the application's ACB must be closed and the MODIFY GR DELETE command must be issued at every host in the sysplex.

Examples of the Display ID=*grname* command are shown in Example 1-105.

Example 1-105 Display ID=GRname shows all members of the GR group

```
D NET,ID=CICSCLP0
   IST097I DISPLAY ACCEPTED
   IST075I NAME = CICSCLP0, TYPE = GENERIC RESOURCE
   IST1359I MEMBER NAME         OWNING CP   SELECTABLE  APPC
   IST1360I USIBMSC.CICSXUU1    CP2NODE        YES       NO
   IST1360I USIBMSC.CICSXUU2    CP4NODE        YES       NO
   IST1360I USIBMSC.CICSXUU0    CP6NODE        YES       NO
   IST314I END

D NET,ID=CICSCLP0,IDTYPE=GENERIC
   IST097I DISPLAY ACCEPTED
   IST075I NAME = CICSCLP0, TYPE = GENERIC RESOURCE
   IST1359I MEMBER NAME         OWNING CP   SELECTABLE  APPC
   IST1360I USIBMSC.CICSXUU1    CP2NODE        YES       NO
   IST1360I USIBMSC.CICSXUU2    CP4NODE        YES       NO
   IST1360I USIBMSC.CICSXUU0    CP6NODE        YES       NO
IST314I END
```

Use Display GRAFFIN to show GR affinities

The DISPLAY GRAFFIN command displays affinity information for generic resources. Since affinities for TSO generic resources exist only temporarily during TSO logon processing, DISPLAY GRAFFIN does not display affinities for sessions with these types of resources.

The display shows partner LUs, generic resource group name, and application member. The results show the partner LUs (whether terminal LUs or application LU6.2 partners) that are in session with participating applications that belong to the generic resource group.

The Display GRAFFIN command shows all GR information in Example 1-106.

Example 1-106 Display GRAFFIN for all GR information

```
D NET,GRAFFIN
    IST097I DISPLAY ACCEPTED
    IST350I DISPLAY TYPE = GENERIC AFFINITY
    IST1706I PARTNER NAME        GENERIC RESOURCE     MEMBER     ATTRIBUTES
    IST1707I USIBMSC.LGTDLU62    USIBMSC.CICSCLP0     CICSXUU2   -VG--SL-
    IST1707I USIBMSC.TST69XPL    USIBMSC.CICSCLP0     CICSXUU1   -VG--WL-
    IST1707I USIBMSC.LGTDLU62    USIBMSC.CICSLUT0     CICSZUPK   -VG--SL-
    IST1707I USIBMSC.PRODLU01    USIBMSC.CICSLUT0     CICSZUPK   -VG--SL-
    IST1707I USIBMSC.LGTDLU62    USIBMSC.CICSCLU0     CICSHGF1   -VG--SL-
    IST1707I USIBMSC.LGTDLU62    USIBMSC.CICSLUT1     CICSPPR3   -VG--SL-
    IST1707I USIBMSC.TST41XPL    USIBMSC.CICSCLU0     CICSHGF1   -VG--WL-
    IST1707I USIBMSC.TST68XPL    USIBMSC.CICSCLP0     CICSXUU0   -VG--WL-
    IST1707I USIBMSC.NVAS005     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVAS006     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVAS007     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0011     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0003     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVAS001     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVASF02     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0007     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVAS002     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0002     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.NVASF00     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVASF01     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.NVASF03     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.NVASF04     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVASF05     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVASF06     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVAS003     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVAS004     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1454I 26 AFFINITIES DISPLAYED
 IST314I END
```

LU specifies the name of the partner LU, as shown in Example 1-107. The name can be a network-qualified name.

Example 1-107 Display GRAFFIN for partner LU

```
D NET,GRAFFIN,LU=PRD0011
    IST097I DISPLAY ACCEPTED
    IST350I DISPLAY TYPE = GENERIC AFFINITY
    IST1706I PARTNER NAME        GENERIC RESOURCE     MEMBER     ATTRIBUTES
    IST1707I USIBMSC.PRD0011     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1454I 1 AFFINITY DISPLAYED
 IST314I END
```

GNAME specifies the generic resource name, as shown in Example 1-108. The name can be a network-qualified name.

Example 1-108 Display GRAFFIN for GR group name

```
D NET,GRAFFIN,GNAME=CICSCLP0
    IST097I DISPLAY ACCEPTED
    IST350I DISPLAY TYPE = GENERIC AFFINITY
    IST1706I PARTNER NAME        GENERIC RESOURCE     MEMBER     ATTRIBUTES
    IST1707I USIBMSC.LGTDLU62    USIBMSC.CICSCLP0     CICSXUU2   -VG--SL-
    IST1707I USIBMSC.TST69XPL    USIBMSC.CICSCLP0     CICSXUU1   -VG--WL-
    IST1707I USIBMSC.TST68XPL    USIBMSC.CICSCLP0     CICSXUU0   -VG--WL-
    IST1707I USIBMSC.NVAS005     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVAS006     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVAS007     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0011     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0003     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVAS001     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVASF02     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0007     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVAS002     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.PRD0002     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.NVASF00     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVASF01     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.NVASF03     USIBMSC.CICSCLP0     CICSXUU2   -VG--W--
    IST1707I USIBMSC.NVASF04     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1707I USIBMSC.NVASF05     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVASF06     USIBMSC.CICSCLP0     CICSXUU0   -VG--W--
    IST1707I USIBMSC.NVAS003     USIBMSC.CICSCLP0     CICSXUU1   -VG--W--
    IST1454I 20 AFFINITIES DISPLAYED
 IST314I END
```

Attributes describes the attributes of the session the affinity represents. *Attributes* is an eight-byte character string that has the following meaning:

1. The first character is either P to indicate that the generic name has been resolved but the session is pending activation, or a hyphen (-) otherwise.
2. The second character is either V to indicate that the affinity will be cleaned up by VTAM when the session is terminated, or A to indicate that the application is responsible for deleting the affinity. Application-owned affinities might not be deleted when the session is terminated. An application-owned affinity will also be deleted when the application is terminated, if the affinity is not persistent (see the fourth attribute character).
3. The third character is either G to indicate that the session was started using the generic name, or A to indicate that the session was started using the application name.
4. The fourth character is either 6 to indicate the affinity will persist after application termination, or a hyphen (-) otherwise. If the affinity will persist it is the responsibility of the application to delete the affinity. An affinity can persist for one of the following reasons:
   - This is an LU6.1 session.
   - This is an LU6.2 SYNCPT session.
   - The application used the SETLOGON GNAMEADD AFFIN=APPL API command to request ownership of all affinities.
   - The application used the LUAFFIN=APPL parameter on the appropriate RAPI or APPCCMD API command to request ownership of this affinity. LUAFFIN=NOTAPPL will override the other three reasons.
5. The fifth character is either M to indicate that the generic resource application also supports MNPS, or a hyphen (-) otherwise. This affinity will not be deleted in a MNPS recovery scenario unless the recovery does not occur before the PSTIMER expires.

6. The sixth character indicates who created the affinity. The value will be one of the following:
   - V to indicate that VTAM resolved the generic name to an application name based on session count or by setting up a session directly to a member of a generic resource group.
   - W to indicate that the Workload Manager resolved the generic name to an application name based on system workload.
   - X to indicate that the generic resource resolution exit resolved the generic name to an application name.
   - S to indicate that the original resolver of the name is not known. This is most likely caused by a previous search request that did not cause an affinity to be created at the time of resolution. This may also occur due to such things as VTAM losing connectivity to the coupling facility, VTAM being re-IPLed, a structure failure of the generic resource coupling facility structure, or following an MNPS recovery.
7. The seventh character is either L to indicate that this affinity will be used for up to 10 minutes after the last session between the session partners ended, or a hyphen (-) otherwise. An affinity might used for this extended time if either of the following is true:
   - An LU6.2session is established without SYNCPT or limited resource.
   - An LU6.2 session is established, but the application specified LUAFFIN=NOTAPPL on the NIB for the OPNDST or OPNSEC.
8. The eighth character is not used and is always a hyphen (-).

Use Modify GR to remove an application from the GR structure

The command is:

```
F NET,GR,GNAME=netid.generic_resource,OPTION=DELETE
```

The MODIFY GR command causes VTAM to delete information about a generic resource locally at this host and from the generic resource's coupling facility structure. If VTAM has lost access to the generic resource's coupling facility structure, local generic resource information can still be deleted.

Note: This command has dependencies that must be met for the command to succeed. See *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781, for complete details on removing generic resource information from the system.

Problem determination for multiple Telnet servers, SD, and GR

Refer to "Problem determination for Telnet server as a standalone task" on page 52.

Refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

If multiple Telnet servers are used (for example, multiple TCP/IP stacks in a SD environment), ensure that each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail or the cross-domain session request will fail.

2

SYSLOGD

The syslog daemon (syslogd) is a server that writes messages from applications to log files. This chapter focuses on syslogd functions that are available in the z/OS V1R8 Communications server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains comprehensive descriptions of the individual parameters for setting up syslogd. It is not the intent of this book to duplicate the information in the manual, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 2.1.2, “For additional information” on page 100.

This chapter discusses the following.

Section	Topic
2.1, “Overview” on page 98	Discusses the basic concepts of syslogd
2.2, “Why syslogd is important” on page 101	Discusses key characteristics of syslogd and why it may be important in your environment
2.3, “The common design scenarios for syslogd” on page 101	Presents commonly implemented syslogd design scenarios, their dependencies, advantages, considerations, and our recommendations
2.4, “How syslogd is implemented” on page 103	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

2.1 Overview

Syslogd is one of the standard applications provided with the z/OS Communications Server. The relationship of syslogd to applications and to other hosts for which it provides logging services is illustrated in Figure 2-1.

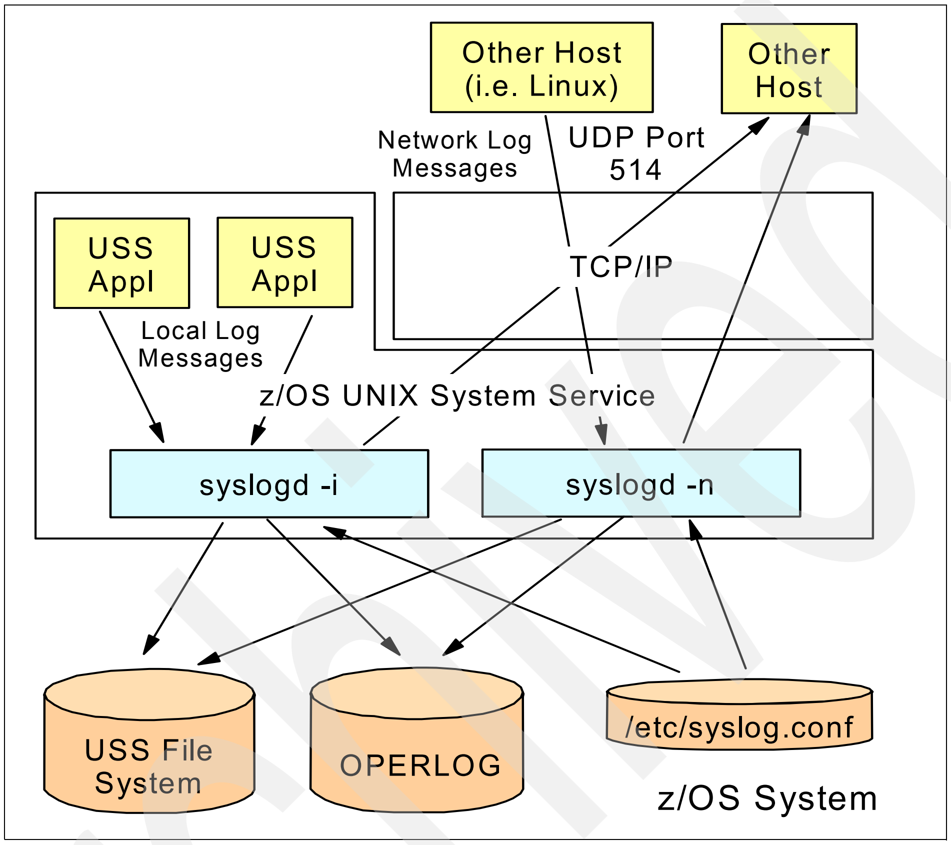


Figure 2-1 syslogd relationship to applications for which it logs information

2.1.1 Basic concepts

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

The syslog daemon receives messages from other systems via 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 only logs messages for applications on that particular system. However, the use of Linux on zSeries® may change this usage when Linux applications are closely related to z/OS applications. This might occur with functions as:

- Middle-tier servers, such as Web Servers or Application Servers
- Communications Controller for Linux (CCL)

In these situations it may be desirable to integrate syslogd messages from these Linux hosts into the z/OS environment. z/OS V1R8 contains enhancements to make this more practical.

The syslog daemon can be started via normal MVS JCL or by a UNIX shell command. It can be stopped by an MVS console operator through a STOP (P) command from the MVS console or from a UNIX shell kill command. When running, the daemon stores its process ID in /etc/syslog.pid or /etc/syslog_net.pid (network-only mode).

When the daemon finishes initialization a message is written to the MVS console indicating the successful start. When the daemon terminates (due to an error condition or due to an operator stop command) a message is sent to the MVS console.

SYSLOGD uses UDP packets for data transfer with the well-known port number 514. RFC 3164 describes the syslog protocol.

syslogd facilities

syslogd uses the concept of facility names to group messages together. Table 2-1 lists these facilities, a definition of each facility, and a list of applications in each facility.

Table 2-1 Facilities

Facility name	Definition	Applications
daemon	Messages generated by generic server applications (daemons)	AT-TLS, ftpd, named, NSLAPM2, oportmap, orexecd, orshd, pagent, pwchange command, pwtokey command, rpcbind, sntpd, osnmpd, syslogd, TCP/IP SNMP subagent, TN3270 SNMP subagent, trmd, trapfwd
auth/authpriv	Messages generated by authorization processes	AT-TLS, orexecd, orshd, otelnetd
user	Messages generated by a generic process	dhcpsd, omproute, tftpd, timed
mail	Messages generated by the mail system	popper, sendmail
news	Messages generated by the news system	No applications shipped by z/OS Communications Server
uucp	Messages generated by the UUCP system	The uucp command
cron	Messages generated by the cron daemon	The cron daemon
lpr	Messages generated by the z/OS UNIX lpr command	The z/OS UNIX lpr command
local0-7	Messages generated by specific local servers.	iked (local4), otelnetd (local1)
mark	Messages generated by syslogd time marks	syslogd

syslogd configuration file

syslogd processing is controlled by a configuration file, usually named /etc/syslog.conf. A sample syslogd configuration file is included in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. 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.

Log messages can be collected from numerous network sources, including Linux hosts, and can be filtered to write to the desired destination based on the source IP address (including subnetwork or the hostname). If you specify /dev/operlog as a direction in the configuration file log messages can be written to the MVS operations log. This provides better performance than writing to /dev/console.

syslogd parameters

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

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

  We highly recommend the use of the -c option. The -c option allows syslogd to 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.
- 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 -i should be specified.
- Network-only mode (-n option)

  You can use the -n option as well. The -n option allows syslogd to receive messages over the network only.
- Disable host name resolution (-x option)

  The -x option causes syslogd to avoid resolver calls for converting IP addresses to hostnames. This option improves performance by avoiding IP address-to-hostname resolution for network log messages.

syslogd isolation

syslogd isolation refers to a particular configuration of syslogd that allows only particular job names to write log messages to particular syslogd facilities and allows the logging of messages received from the local host only or over the network only. The syslogd isolation feature is automatically enabled when job names are included as a filter option in the syslogd configuration file and when the -i or -n syslogd option is specified at startup.

2.1.2 For additional information

For additional information, refer to:

- *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 "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

2.2 Why syslogd is important

Many applications included in the z/OS Communications Server, other IBM applications, and third-party applications will log messages via syslogd. In order to organize and keep track of these messages, it is very important to configure syslogd.

2.3 The common design scenarios for syslogd

In this section we discuss the common design scenarios for syslogd.

2.3.1 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 every day.

Dependencies

Prior to starting, syslogd requires an /etc/syslog.conf configuration file containing a single logging directive. In order to tell syslogd to switch to a new file name, a cron job needs to be created to send a signal to syslogd informing it that a new day has started and that it should reread the configuration file.

Advantages

This configuration can be set up in a matter of minutes and is easy to understand.

Considerations

Sending all messages to a single file makes it difficult to follow what is occurring in a single application over many hours.

2.3.2 Sending messages to different files based on job name

In this section we discuss a syslogd configuration where each application sends messages to a syslogd different 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.

Dependencies

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

Advantages

This configuration makes it very 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.

Considerations

This configuration makes it difficult to determine what is occurring system wide across all applications at a given point in time. This configuration also requires each job name to be specified in the syslog.conf. Any job name not explicitly coded will not be saved by syslogd.

2.3.3 Sending messages to different files and to a single file

In this section, we discuss combining the previous scenarios into one configuration.

Dependencies

Prior to starting, syslogd requires an /etc/syslog.conf configuration file containing multiple logging directives, one per application, and one additional directive identifying the file that will log all message in a single location. In order to use file management scripts, you will need cron configured and running. Cron is documented in *z/OS UNIX System Services Planning*, GA22-7800.

Advantages

This configuration makes it very easy to find messages related to a particular application and follow the flow of events. It also provides a complete picture of what is occurring on the system at a given point in time.

Considerations

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, at a minimum, end up in the file that is logging all messages.

2.3.4 Starting two instances of syslogd

In this section, we discuss starting two instances of syslogd. One of the two instances receives log messages locally, and the other receives log messages from other hosts such as Linux on zSeries.

Dependencies

Prior to starting, two instances of syslogd requires an /etc/syslog.conf configuration file. It contains multiple logging directives, one per application, and /dev/operlog as a logging directive, one per hostname or IP address, and one additional directive identifying the file that will log all specific network message in a single location.

Advantages

This syslogd Isolation helps ensure that local syslogd logging is not adversely affected by the amount of remote messages being forwarded to z/OS. These configurations make it very easy to find messages related to a particular application and follow the flow of events. It also provides a complete picture of what is occurring on the system and other hosts at a given point in time.

Considerations

Network syslogd messages are delivered via UDP. UDP does not guarantee the delivery of messages; moreover, messages may not be delivered to z/OS under some conditions. IPSEC should be considered for protecting the syslog daemon's UDP port 514 when running in normal or network-only mode. For more information, see *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

2.3.5 Recommendations

We recommend configuring syslogd to log to individual files based on job name and to also simultaneously log all messages in a single file. By using this design we benefit from the advantages of each configuration and offset the considerations if you use a local-only mode syslogd.

If you use both local and network logging, we recommend that you use two instances of syslogd. By using the scenario introduced in 2.3.4, “Starting two instances of syslogd” on page 102, we benefit from the advantages of each configuration and offset the considerations.

2.4 How syslogd is implemented

Based upon our recommendations from 2.3.5, “Recommendations” on page 103, we show two implementations in detail.

- Sending messages to different files and to a single file.
- Starting two instances of syslogd

2.4.1 Sending messages to different files and to a single file

In this section we discuss the implementation tasks necessary to set up and run syslogd using a configuration that logs to different files based on job name and also logs each message to a single common file. In order to minimize DASD usage, we also discuss management techniques to prevent the z/OS UNIX file system from filling with outdated log files.

Implementation tasks

These tasks are:

1. Create an /etc/syslog.conf configuration file.
2. Create a crontab entry for syslogd and the management scripts.
3. Start syslogd.

Create an /etc/syslog.conf configuration file

We recommend copying the /usr/lpp/tcpip/samples/syslog.conf to /etc/syslog.conf as a starting point. Copying of the sample configuration file is not required for our example shown in Example 2-1 to work, but will add a large number of comments to the beginning of the file that explains the syntax of the file. If you choose to copy the sample, then delete the last four lines of the configuration file, starting from the line that reads `THIS EXAMPLE STATEMENT IS UNCOMMENTED` to the end of file. Then add the lines shown in Example 2-1.

Example 2-1 Our /etc/syslog.conf configuration file

```
*.INETD*.*.* /var/log/%Y/%m/%d/inetd
*.FTP*.*.* /var/log/%Y/%m/%d/ftp
*.SYSLOGD*.*.* /var/log/%Y/%m/%d/syslogd
*.OMPR*.*.* /var/log/%Y/%m/%d/omproute
*.SENDMAIL.*.* /var/log/%Y/%m/%d/sendmail
*.NAMED*.*.* /var/log/%Y/%m/%d/named
# Add additional jobnames here as needed

# Send a copy of all message, regardless of jobname, to the file named msgs
*.* /var/log/%Y/%m/%d/msgs
```

Create a crontab entry for syslogd

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

Important: Cron must be configured and running in order to automatically run these scripts. Cron is documented in *z/OS UNIX System Services Planning*, GA22-7800.

Example 2-2 Crontab entries

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

Start syslogd

Add the lines in Example 2-3 to /etc/rc to start syslogd every time z/OS UNIX starts.

Example 2-3 /etc/rc entry to start syslogd

```
export _BPX_JOBNAME='syslogd'
/usr/sbin/syslogd -c -i &
```

Alternatively, you can use the started task in Example 2-4 to start syslogd.

Example 2-4 Started task for SYSLOGD

```
//SYSLOGD PROC
//SYSLOGD EXEC PGM=SYSLOGD,REGION=0K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)',
//       'ENVAR("TZ=EST5EDT")/')
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Verification

To verify that syslogd is working correctly, go to the directory that contains the log files and check the log files. Example 2-5 shows the output of **`/bin/ls`** that was issued while in our /var/log/2006/10/23 directory.

Example 2-5 Output of /bin/ls in our log directory

```
# cd /var/log/2006/10/23
# ls -l
total 48
-rw-------   1 IBMUSER  SYS1          0 Oct 23 21:04 ftp
-rw-------   1 IBMUSER  SYS1         86 Oct 23 21:12 inetd
-rw-------   1 IBMUSER  SYS1       9529 Oct 23 21:12 msgs
-rw-------   1 IBMUSER  SYS1          0 Oct 23 21:04 named
-rw-------   1 IBMUSER  SYS1       4468 Oct 23 21:10 omproute
-rw-------   1 IBMUSER  SYS1          0 Oct 23 21:09 sendmail
-rw-------   1 IBMUSER  SYS1          0 Oct 23 21:04 syslogd
```

Example 2-6 shows the contents of one of our log files.

Example 2-6 Contents of the omproute log file

```
Oct 24 00:10:21 OMPROUTE omproute[83886100]: EZZ7800I OMPROUTE starting
Oct 24 00:10:22 OMPROUTE omproute[83886100]: EZZ7845I Established affinity with TCPCS
Oct 24 00:10:22 OMPROUTE omproute[83886100]: EZZ7817I Using default OSPF protocol 89
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7817I Using default OSPF protocol 89
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7838I Using configuration file:
/tmp/omproute/omproute-marc.cf
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7890I Non-broadcast ignored when
OSPF_Interface statement is a wildcard
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 162.139.20.129, name VLINK, index 0, flags 4041
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 162.139.20.250, name VLINK2, index 0, flags 4041
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 9.42.105.88, name ETH1, index 1, flags 441
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 172.21.0.1, name CTC12, index 2, flags 450
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 10.10.1.3, name CTC13, index 3, flags 450
Oct 24 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 9.9.9.31, name EZASAMEMVS, index 4, flags ffff8c50
```

Problem determination

If a log file does not contain any messages, first make sure the job is running. If the job is running, restart syslogd and add -u as an additional parameter. The -u parameter adds user ID/job name information to the message. Once syslogd is running again with the -u parameter, check the msgs file to determine if the job-specific log file is empty because the job name does not match the job name specified in /etc/syslog.conf. Example 2-7 shows a section of our msgs file where INETD and syslogd were started with the wrong job name.

Example 2-7 Entries in msgs file with incorrect job name

```
Oct 23 21:07:45 MVS130/CS01 1  CS019 2 inetd[19]: FOMN0066 inetd terminating
Oct 23 21:09:33 MVS130/CS01 1  CS016 3 FSUM1220 syslogd: restart
Oct 23 21:09:33 MVS130/CS01 1  CS016 3 FSUM1232 syslogd: running non-swappable
```

The field immediately following the date appears as a a result of the -u parameter and identifies the system/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** **3**. The next field after the job name is the beginning of the message. As you can see in Example 2-7, inetd was incorrectly named CS019 **2** and syslogd was incorrectly named CS016 **3**, which would explain why Example 2-7 shows that the inetd and syslogd log files are empty.

For other problems, start syslogd with the -d parameter to enable a debug trace. Output from the debug trace will be sent to the stdout stream.

2.4.2 Starting two instances of syslogd

In this section we discuss the implementation tasks necessary to set up and run two instances of syslogd. It consists of tasks based on 2.4.1, “Sending messages to different files and to a single file” on page 103. Our environment has a z/OS Communications Server and a Linux server.

Implementation tasks

These tasks are:

1. Create /etc/syslog.conf configuration files.
2. Create crontab entries for syslogd and the management scripts.
3. Start syslogd.

Create /etc/syslog.conf configuration files

We configured only one configuration file to be shared by two instances of syslogd. Example 2-8 shows our configuration file.

Tip: If you intend to use "*.*" as a filter, you should configure two separate configuration files for each syslogd, because this wildcard "*.*" actually matches all hostnames and IP addresses. At this time, network-only mode syslogd does not send its log messages to another syslogd, therefore, you should define "syslogd*" as jobname in a configuration file of network-only mode syslogd if you want to collect log messages of network-only mode syslogd as well.

Example 2-8 /etc/syslog.conf configuration file for z/OS

```
###########################
#   For local-only mode   #
###########################
*.INETD*.*.* /var/log/%Y/%m/%d/inetd
*.FTP*.*.* /var/log/%Y/%m/%d/ftp
*.SYSLOGD*.*.* /var/log/%Y/%m/%d/syslogd
*.OMP*.*.* /var/log/%Y/%m/%d/omproute
*.SENDMAIL.*.* /var/log/%Y/%m/%d/sendmail
*.NAMED*.*.* /var/log/%Y/%m/%d/named
# Add additional jobnames here as needed

###########################
#  For network-only mode  #
###########################
(10.30.2.99).daemon.debug /dev/operlog 1
# Add additional hostspecs here as needed

# Send a copy of all messages to the file named msgs_net
(10.30.2.99).*.* /var/log/%Y/%m/%d/msgs 2
```

- **1** Log messages from 10.30.2.99 can be written to /dev/operlog
- **2** We defined IP address 10.30.2.99 to prevent syslogd from receiving messages from unintended hosts.

We added an entry **1**, as shown in Example 2-9, into the /etc/syslog.conf configuration file on Linux, and then restarted the syslog daemon on it.

Example 2-9 /etc/syslog.conf entry on Linux

```
*.*                                 @10.20.1.221 1
```

Create crontab entries for syslogd

From the z/OS UNIX shell issue the command **`export EDITOR=oedit`**, and then issue the command **`crontab -e`**. Using the editor, add the two lines shown in Example 2-10. Save the file and exit the editor. These entries allow log files to be recreated daily.

Example 2-10 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 syslog daemon stores its process ID in the /etc/syslog_net.pid in addition to the /etc/syslog.pid.

Start syslogd

We issued z/OS UNIX shell commands as follows:

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

To start syslogd every time z/OS UNIX starts we added the same lines to /etc/rc as we issued, as shown in Example 2-11.

Example 2-11 /etc/rc entries to start syslogd

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

Verification

To verify that the syslog daemons are working correctly, we can use a **ps -ef | grep syslogd** command and verify the log files in the directory match the defined configuration files. Example 2-12 shows a process list. Example 2-13 shows the output of a **ls** command that was issued while in the /var/log/2006/08/24 directory.

Example 2-12 Process list

```
BPXROOT @ SC32:/>ps -ef | grep syslogd
 BPXROOT   84082918          1  - 16:21:05 ?        0:00 /usr/sbin/syslogd -f /
etc/syslog.conf -c -n -x 1
 BPXROOT   33751271          1  - 16:20:53 ?        0:00 /usr/sbin/syslogd -f /
etc/syslog.conf -c -i 2
```

- 1 shows the network-only mode syslog daemon.
- 2 shows the local-only mode syslog daemon.

Example 2-13 Output of the ls command

```
BPXROOT @ SC32:/>cd /var/log/2006/08/24
BPXROOT @ SC32:/SC32/var/log/2006/08/24>ls -l
total 40
-rw-------   1 BPXROOT  SYS1          0 Aug 24 16:20 ftp
-rw-------   1 BPXROOT  SYS1          0 Aug 24 16:20 inetd
-rw-------   1 BPXROOT  SYS1       9525 Aug 24 17:19 msgs
-rw-------   1 BPXROOT  SYS1          0 Aug 24 16:20 named
-rw-------   1 BPXROOT  SYS1       1530 Aug 24 17:53 omproute
-rw-------   1 BPXROOT  SYS1          0 Aug 24 16:20 sendmail
-rw-------   1 BPXROOT  SYS1       1558 Aug 24 17:52 syslogd
```

Example 2-14 shows the output of /dev/operlog. The *REMOTE* (1) shows that this message was received from a remote host, and the following message (2) illustrates a log message from a syslog daemon on a server which has an IP address of 10.30.2.99.

Example 2-14 /dev/operlog

```
/dev/operlog
```

```
M 0000000 *REMOTE*1 2006236 16:42:23.17 STC00070 00000280  BPXF060I LOGGED BY SYSLOGD FROM
A REMOTE SOURCE     000
D                                      000 00000280  Aug 24 16:42:23 ::ffff:10.30.2.99
namedÝ5574¨: starting BIND 9.2.4 -u 2
E                                      000 00000280  named
M 0000000 *REMOTE* 2006236 16:42:23.17 STC00070 00000280  BPXF060I LOGGED BY SYSLOGD FROM A
REMOTE SOURCE     000
E                                      000 00000280  Aug 24 16:42:23 ::ffff:10.30.2.99
namedÝ5574¨: using 2 CPUs
M 0000000 *REMOTE* 2006236 16:42:23.18 STC00070 00000280  BPXF060I LOGGED BY SYSLOGD FROM A
REMOTE SOURCE     000
D                                      000 00000280  Aug 24 16:42:23 ::ffff:10.30.2.99
namedÝ5574¨: loading configuration
E                           000 00000280  from '/etc/named.conf'
M 0000000 *REMOTE* 2006236 16:42:23.19 STC00070 00000280  BPXF060I LOGGED BY SYSLOGD FROM A
REMOTE SOURCE     000
D                                      000 00000280  Aug 24 16:42:23 ::ffff:10.30.2.99
namedÝ5574¨: listening on IPv4
E                           000 00000280  interface lo, 127.0.0.1#53
M 0000000 *REMOTE* 2006236 16:42:23.20 STC00070 00000280  BPXF060I LOGGED BY SYSLOGD FROM A
REMOTE SOURCE     000
D                                      000 00000280  Aug 24 16:42:23 ::ffff:10.30.2.99
namedÝ5574¨: listening on IPv4
E                                      000 00000280  interface eth0, 10.30.2.99#53
```

Example 2-15 shows the content of syslogd log file.

Example 2-15 syslogd log file

```
/var/log/2006/08/24/syslogd
BPXROOT @ SC32:/SC32/var/log/2006/08/24>cat syslogd
Aug 24 16:20:52 WTSC32 syslogd: FSUM1220 syslogd: restart
Aug 24 16:20:52 WTSC32 syslogd: FSUM1237 Job SYSLOGD3 running in local-only mode 1
Aug 24 16:20:52 WTSC32 syslogd: FSUM1232 syslogd: running non-swappable
Aug 24 16:21:04 WTSC32 syslogd: FSUM1220 syslogd: restart
Aug 24 16:21:04 WTSC32 syslogd: FSUM1238 Job SYSLOGD4 running in network-only mo 2
de
Aug 24 16:21:04 WTSC32 syslogd: FSUM1232 syslogd: running non-swappable
```

Key log messages are as follows:

- 1 shows Job SYSLOGD3 running in local-only mode.
- 2 shows Job SYSLOGD4 running in network-only mode.

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. This chapter focuses on the FTP functions that are available in the z/OS V1R8 Communications server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, contain comprehensive descriptions of the individual parameters for setting up FTP. They also include step-by-step checklists and supporting examples. It is not the intent of this book to duplicate the information in the manuals, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 3.1.7, "Additional information sources" on page 124.

This chapter discusses the following.

Section	Topic
3.1, "Overview" on page 110.	Discusses the basic concepts of FTP
3.2, "Why FTP is important" on page 124.	Discusses key characteristics of the FTP and why it may be important in your environment
3.3, "Common design choices for FTP" on page 125.	Presents commonly implemented FTP design scenarios, their dependencies, advantages, considerations, and our recommendations
3.4, "How FTP is implemented" on page 131.	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

3.1 Overview

File Transfer Protocol is the name of the UNIX application and the protocol that allows you to transfer files in a TCP/IP network. 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.

There are two machines involved in an FTP session, namely the local host, which is the client machine, and a remote host, which 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. This host 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.

As illustrated in Figure 3-1, FTP is one of the standard applications provided with the z/OS Communications Server.

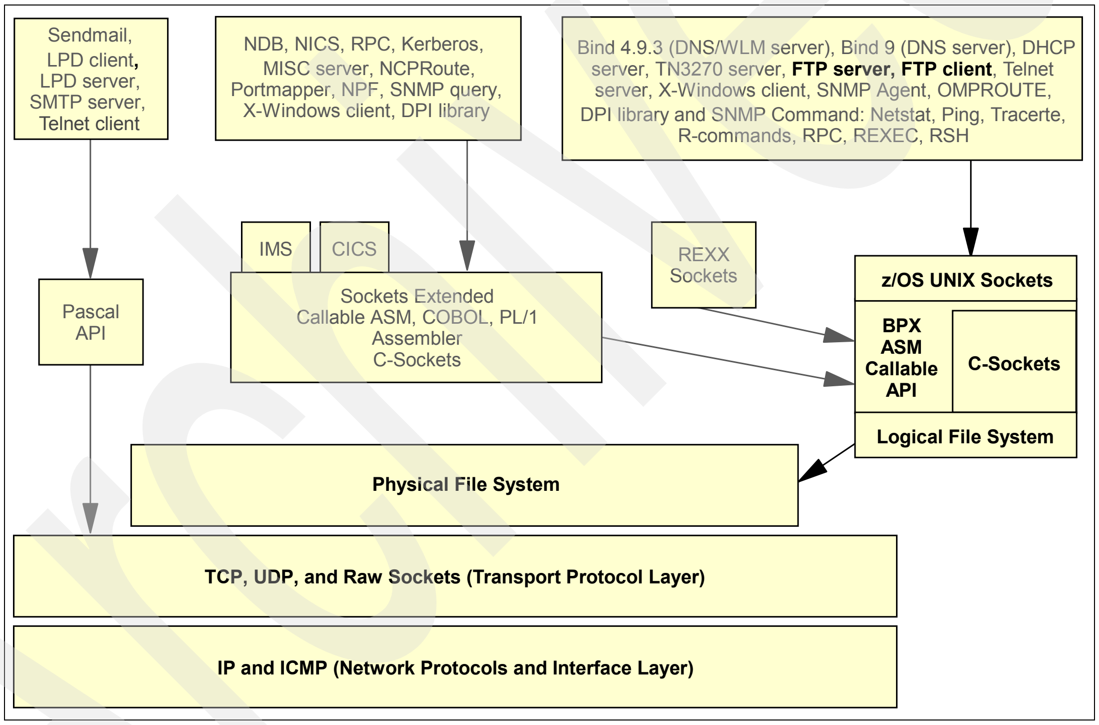

Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270 server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
REXX Sockets
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
z/OS UNIX Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

Figure 3-1 z/OS FTP client/server application services

3.1.1 Basic concepts

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. Refer to *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.

The relationship between the FTP client and FTP server is shown in Figure 3-2.

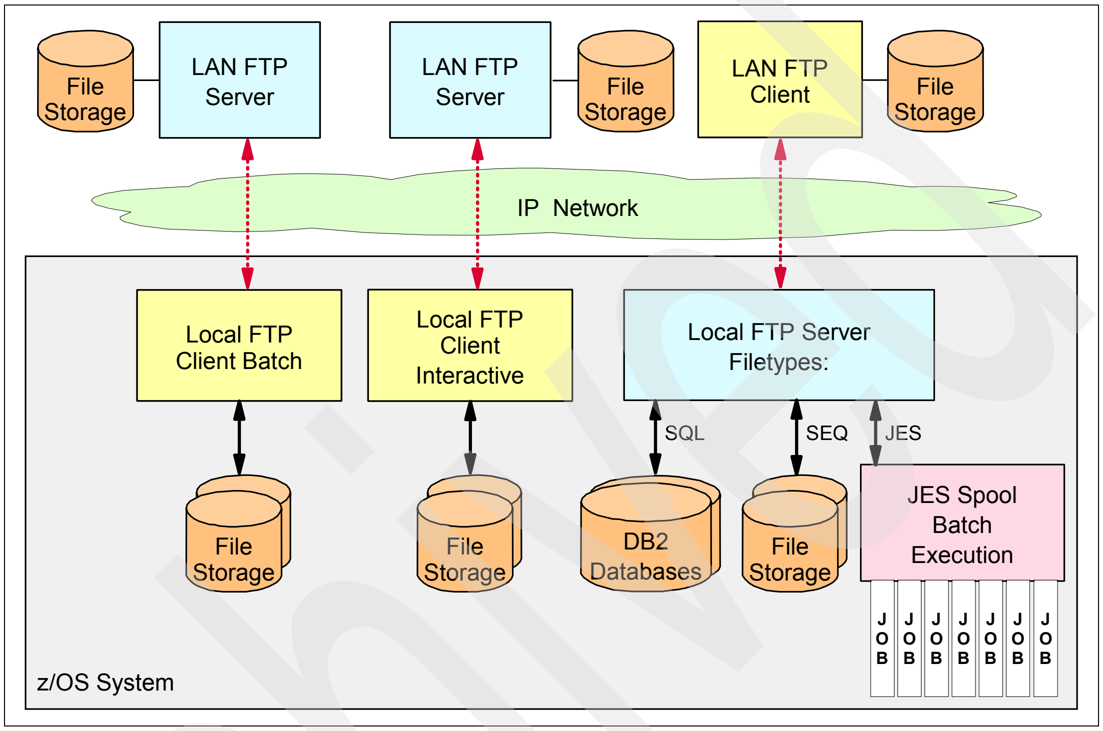


Figure 3-2 FTP client/server relationships

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 UNIX System Services 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 both 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.
- Access control via support for user exits.
- 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).
- The client can be invoked from TSO, from the UNIX system services shell, as a batch job, or by an application programming interface (API).
- Support for FTP proxy connections via a SOCKS server.
- Controls in FTP.DATA for server and client operations.

- Support for UNICODE UTF-8 encoding.

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(FTPCDATA). *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. At all subsequent forks, a new job name is chosen in one of two ways:

- BPX_JOBNAME
- Assigned by z/OS UNIX

BPX_JOBNAME

The BPX_JOBNAME parameter can be specified in the started procedure. If specified, all forked job names use the specified name.

Assigned by z/OS UNIX

If the original job name is less than eight characters, 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 initial job name is FTPD. At each fork, the job name with be FTPD*x,* where *x* is a number between one and nine. If the initial job name is already eight characters long, then the job name does not change across forks.

3.1.2 Enhancements in FTP server and client in using FTP.DATA dataset

There are two options have been added to z/OS Communications Server to improve choices for facilitating information and configuration management.

Option to change server reply code from 250 to 226

This option 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 proxy 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 in order to comply with the RFC standard. This change is being implemented via 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:

- **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 command level allows the FTP client to specify a dataset, HFS file, or ddname to override the FTP.DATA search order. This option is implemented via an FTP.DATA parameter to allow customer control over its usage.

The addition of the -f parameter changes the FTP.DATA search order:

Table 3-1 Option to set the FTP.DATA 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) dataset 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

Important: It is important to specify a correct ftp.data file with the -f parameter. If the specified file is not found, the standard ftp.data order is searched. No error message is issued.

3.1.3 Platform-specific features via SITE/LOCSITE commands

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. In order 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.

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 in order 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 may need. The client can change many of these values for the current FTP session via 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 may change these via a LOCSITE command.

You do not necessarily need to specify all the allocation attributes of an MVS data set; you may 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:

- 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 way 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 may contain data sets or subdirectories. A partitioned data set is considered a directory, and the individual members as files in that directory. You may step down the hierarchy by using **CD** commands to name the next low-level qualifier you want to view. You may 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 may 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, just to mention the most obvious. Some operating systems organize files into records, while 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 in order 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 via 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 in order 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, neither at the sending nor 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*.

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. This is a relatively simple matter to deal with as long as we talk about text files. The ASCII data type with CRLF sequences can be seen as a case of record data. 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', in order 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 via 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 may use block mode when you transfer files between S/370™ 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 above, you have to 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 very 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 may not need to translate between EBCDIC and ASCII, and you can use the BINARY data type instead.

3.1.4 FTP LOCSTat and STAtus subcommands enhancements

In z/OS Communication Server V1R8, the FTP client subcommands LOCSTAT and STATUS have the ability to display status selectively. This is accomplished with the use of a parameter. The LOCSTAT subcommand displays the output of *client* status information, and the STAT subcommand displays the status of parameters in the *server*.

Restriction: Only one argument can be used at a time with the LOCSTAT and STAT subcommands. An example of the stat command is shown in Example 3-3.

Example 3-3 Use of STAT and LOCSTAT commands.

```
ftp> quote stat (autor
211-Automatic recall of migrated data sets.
211 *** end of status ***
ftp>
EZA1460I Command:
EZA1736I   STAT (INACTIVETIME
EZA1701I >>> XSTA (INACTIVETIME
211-Inactivity timer is set to 300
211 *** end of status ***
EZA1460I Command:
EZA1736I   LOCSTAT DCONNTIME
EZA2811I DCONNTIME is 120
EZA1460I Command:
```

3.1.5 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 may 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

[1] Including all the descendants of DOS.

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 since 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.

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 may 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 in the following way:

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 may 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

To do this:

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 se 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

To do this:

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 may 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. Since 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.

While the technique above 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 easily transfer any VB files via structure mode between MVS systems.

Transferring a recfm=vb data set between z/OS systems

To do this:

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. Since 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.

3.1.6 FTP UTF-8 data transfer and storage

z/OS V1R8 Communication Server introduced FTP support of UNICODE file transfers. UNICODE provides a unique number for every character, no matter what platform, program or language you are using.The same code page (UNICODE) will be supported regardless of which operating system it runs on (z/OS, Windows®, Linux, AIX®, etc.). 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 initial implementation of UNICODE is a step in moving z/OS to fuller Unicode enablement.

Considerations

For our usage we needed to observe the following points:

- The only UNICODE encoding supported for z/OS FTP file transfer is UTF-8.
- 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, as well as the use of 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”. It refers to the byte order in which multiple number 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

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 new line 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 may be coded in the z/OS FTP server FTP.DATA file, the z/OS FTP client FTP.DATA file, or issued via 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:

- 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, as well as 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.

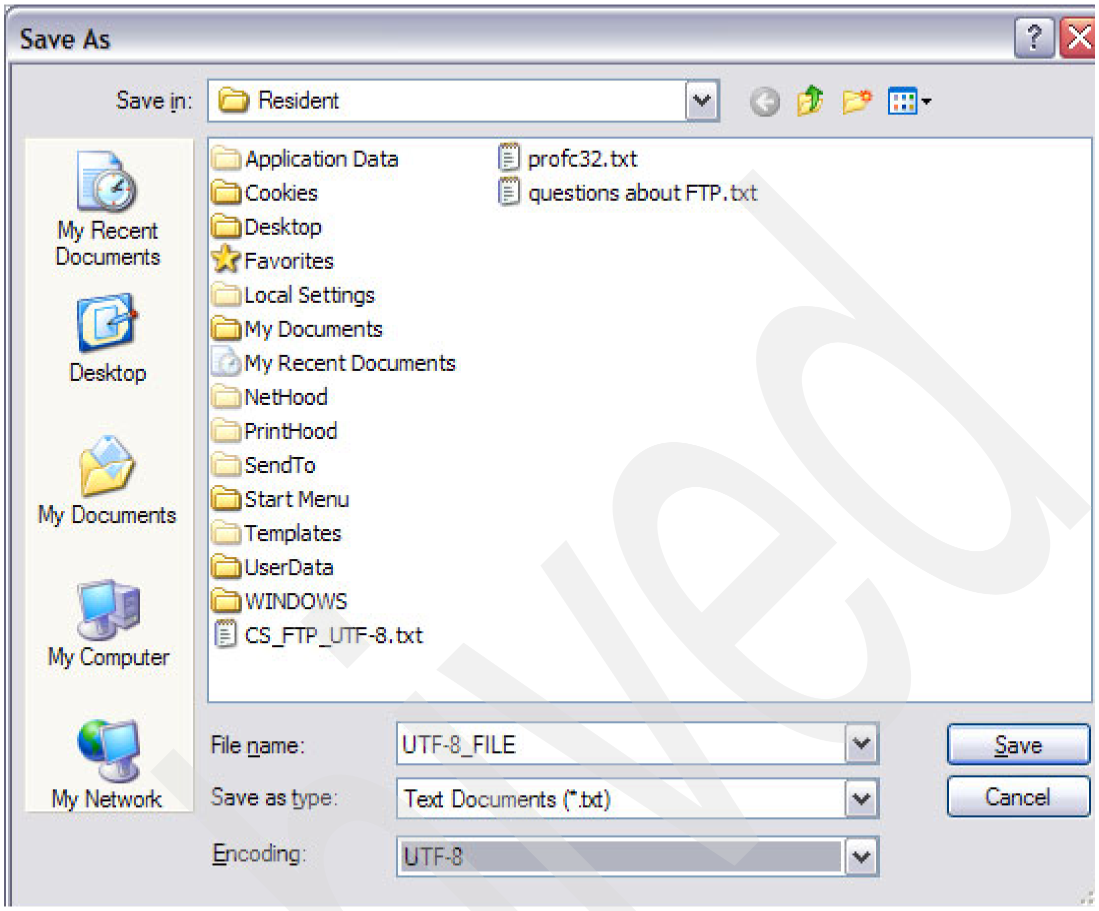
MBDATACONN is used to define the conversions between file system code page and the network transfer code page during data transfer. It affects the conversion of double-byte character set (DBCS) and multibyte character set (MBCS) data and is used when the ENCODING MBCS statement is coded. The SITE and LOCSITE subcommands are also available to set this keyword. The options (UTF-8,UTF-8) are used to support UNICODE encoding.

Example 3-4 FTP.DATA with parameters and options for UNICODE.

```
Encoding        MBCS          ;
MBdataconn  (UTF-8,UTF-8)  ;
Unicodefilesystembom ASIS  ;
MBrequirelastEOL TRUE      ;
```

In Example 3-4 we illustrate the use of these parameters. This example is from a SYSFTPD dataset issued to store a file on the server.

To create UTF-8 file on our workstations we simply used the Windows notepad to edit a file and save it with the appropriate encoding option, as shown in Figure 3-3. Example 3-5 illustrates the complete FTP dialog for the file.



File name: UTF-8_FILE
Save as type: Text Documents (*.txt)
Encoding: UTF-8

Figure 3-3

Example 3-5 FTP from the Windows client to z/OS FTP Server

```
C:\>ftp 10.40.2.232
Connected to 10.40.2.232.
220-FTPDD1 IBM FTP CS V1R8 at WTSC30D.itso.ibm.com, 18:03:14 on 2006-08-10.
220 Connection will close if idle for more than 5 minutes.
User (10.40.2.232:(none)): cs06
331 Send password please.
Password:
230 CS06 is logged on.  Working directory is "CS06.".
ftp> ascii
200 Representation type is Ascii NonPrint
ftp> quote site encoding=mbcs
200 SITE command was accepted
ftp> quote site mbdataconn=(utf-8,utf-8)
200 SITE command was accepted
ftp> quote site unicodefilesystembom=asis
200 SITE command was accepted
ftp> put c:\UTF-8_FILE.txt 'TCPIP.ITSO.FTPUTF8'
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
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>
```

Dependencies

When a MBCS (multibyte character set) 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.

```
451 Transfer aborted due to file error. File is catalogued.

ftp: 125 bytes sent in 0.00Seconds 125000.00Kbytes/sec.

ftp> put c:\CSFTPUTF 'TCPIP.ITSO.FTPUTF8'

200 Port request OK.

125 Storing data set TCPIP.ITSO.FTPUTF8

451-File Transfer might be incomplete. Last record received without EOL sequence
```

It would appear the file transfer failed, but actually the file has been stored in the dataset. 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 and 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.

Attention: As this is an optional feature, there are no migration issues. To take advantage of these function you need only use the technique of specifying Unicode operation via MBDATACONN, MBREQUIRELASTEOL and UNICODEFILESYSTEMBOM. These can be used through the SITE and LOCSITE commands, without changing FTP.DATA.

3.1.7 Additional information sources

For additional information, refer to:

- *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.

3.2 Why FTP is important

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.

3.3 Common design choices for FTP

The following are common design choices for implementing FTP:

- FTP without security
- FTP with security
- Multiple FTP server instances in a multiple stack environment
- Single generic FTP server in a multiple stack environment
- Load balancing across FTP servers in a sysplex
- FTP client with SOCKS proxy protocol security
- FTP client via batch
- FTP client via the FTP client API
- FTP management with SMF

3.3.1 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.

Dependencies

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. Starting syslogd prior to starting FTP is highly recommended for retaining messages from the FTP server.

Advantages

FTP running without security requires less overhead than FTP running with security.

Considerations

Without using z/OS Communications Server security capabilities, you may need to implement necessary security elsewhere (in your firewalls or routers, for example) or you may elect to transmit your data in the clear.

3.3.2 FTP with security

In this section we discuss the security parameters to the FTP design described in 3.3.1, "FTP without security" on page 125. We need to enable various optional RACF profiles and then authenticate and encrypt the FTP sessions. The FTP server and client in the z/OS Communications Server supports TLS or Kerberos security. We recommend using TLS.

Note: FTP can also be made secure with z/OS Communications Server Application Transparent TLS (AT-TLS) or IP Security (IPSec). This section only discusses the TLS security built into FTP. AT-TLS and IPSec are discussed in *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

Implicit versus explicit TLS connections

An implicit TLS connection is one in which the client connects to a nonstandard port and immediately performs TLS negotiation. An explicit TLS connection is one where the client connects to the standard FTP server port (21) and immediately issues a command indicating that TLS should be enabled. In our implementation of secure FTP in 3.4.2, “FTP with security” on page 137, we demonstrate the use of explicit TLS.

Dependencies

In addition to the dependencies in 3.3.1, “FTP without security” on page 125, FTP with security requires additional statements in FTP.DATA and a key ring database with at least one certificate.

Advantages

FTP with security provides a safer environment for the transmission of data.

Considerations

The use of security requires more configuration and requires management of key rings. The use of TLS also adds overhead to the FTP transfer and requires an FTP client that supports TLS.

3.3.3 Multiple FTP server instances in a multiple stack environment

In this section we discuss running multiple instances of the FTP server in a multiple stack environment.

Dependencies

Multiple stacks and multiple FTP servers must be configured.

Advantages

If you need two FTP servers that will be configured differently from each other, and want to run both FTP servers on a single system, then using multiple FTP servers on multiple stacks is one way to accomplish this goal. For example, if you want one LPAR to provide both a secure and non-secure FTP server, then you could set up a basic FTP server to listen on one stack and another FTP server with security configured to listen on a different stack that is configured to use some of the security techniques discussed in *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

You can use the _BPXK_SETIBMOPT_TRANSPORT environment variable to associate one FTP server with one stack. In each FTP server’s started procedure, you need to specify the job name of the stack that this particular FTP server uses. For example, if you had an FTP server with a job name of FTPDA and a TCP/IP stack of TCPIPA, you would specify _BPXK_SETIBMOPT_TRANSPORT=TCPIPA in the started task for FTPDA.

Considerations

Each FTP server must be configured individually on separate stacks. Each FTP server needs a separate address space. This results in many additional address spaces and higher CPU load.

The use of multiple TCP/IP stacks is not recommended. Using FTP across multiple LPARs on multiple systems will provide the same benefits as using FTP on multiple stacks. For example, setting up one sysplex to use secure stacks and secure FTP servers and setting up a different a different sysplex to use stacks and FTP servers without security is a more recommended approach with a number of additional benefits.

3.3.4 A single generic FTP server in a multiple stack environment

In this section we discuss starting a single FTP server to handle connections from multiple stacks.

Dependencies

Multiple stacks must be configured and active.

Advantages

If you need two stacks configured differently but only one FTP server, then this configuration is one way to accomplish that goal. For example, you could have one stack with a firewall and policies active, and one stack without a firewall and policies. This scenario differs from that discussed in 3.3.3, “Multiple FTP server instances in a multiple stack environment” on page 126, in that only the stacks are different. One FTP configuration is capable of meeting the needs of either stack configuration. This configuration has an additional advantage because only one FTP instance needs to be configured.

Considerations

The same considerations mentioned in 3.3.3, “Multiple FTP server instances in a multiple stack environment” on page 126, apply to this configuration as well. Additional stacks require additional resources in terms of CPU load and memory load. The same configuration is best accomplished using multiple LPARs and sysplexes.

3.3.5 Load balancing FTP servers in a sysplex

This section provides an overview of executing multiple FTP servers and using SD to load balance between them. Based upon installation policies, the SD directs connections to the best available FTP server.

For this situation we use two system images, each with one TCP/IP stack and one FTP server associated with it. The TCP/IP stack started task name is the same on both systems: TCPIPB. The FTPD server started task name is the same on both systems: FTPDB. We use system symbolics in the started task JCL to provide uniqueness where necessary.

On system SC30 we have:

- TCPIPB
- FTPDB

On system SC31 we have:

- TCPIPB
- FTPDB

We designed the two stacks involved to back each other up. We also designed the two FTP servers to back each other up. 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

In order to use FTP in a SD environment, you need to configure FTP on each LPAR in the sysplex using the directions discussed in 3.3.1, “FTP without security” on page 125, or in 3.3.2, “FTP with security” on page 125, depending on whether you want to use secure FTP.

Stack dependencies for multiple FTPD servers with SD

Because sysplex distribution (SD) is being used in this scenario, all the functionality that a TCP/IP stack needs to support SD is required. Refer to *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341, for a discussion on setting up a TCP/IP stack to support sysplex distribution. These requirements include:

- 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 use 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 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. Our scenario calls for an identical process.

The introduction of sysplex distribution adds the requirement for a new IP subnet for Distributed DVIP as though Dynamic VIPA has not already been implemented.

FTP server dependencies for multiple servers with SD

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

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, refer to those discussions in the following resources:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Considerations

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 what the physical network requirements are.

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 may be required. These issues are sometimes overlooked.

3.3.6 FTP client with SOCKS proxy protocol security

In this section we discuss the configuration changes necessary to add SOCKS server support to the FTP client we introduced in 3.3.1, “FTP without security” on page 125.

Dependencies

In addition to the dependencies discussed in 3.3.1, “FTP without security” on page 125, the FTP client with SOCKS requires a SOCKS server configuration file and an additional statement in FTP.DATA identifying the SOCKS server configuration file.

Advantages

Use of the FTP client with SOCKS allows users to contact FTP servers that are protected by a SOCKS firewall.

3.3.7 FTP client via batch

In this section we discuss running the FTP client as a batch job.

Dependencies

In addition to the dependencies discussed in 3.3.1, “FTP without security” on page 125, an FTP batch job requires JCL containing the FTP commands to be executed.

Advantages

Submitting batch FTP jobs allows non-interactive FTP sessions that run in the background.

Considerations

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.

3.3.8 FTP client via the FTP client API - Rexx support

In this section we discuss the use of the FTP client API Rexx support.

In previous releases, FTP client already had an interface through the FTP client API for COBOL, C, Assembler and PL/I. z/OS V1R8 Communication Server extends the FTP client programming interface support by providing a Rexx API.

Dependencies

An application using the FTP client API must be written in COBOL, C, Assembler, REXX or PL/I. The *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787, lists additional requirements of the application.

Advantages

The FTP client API allows any application to directly invoke the functionality of the FTP client running on z/OS with significantly improved automation capabilities for file transfer operations. Rexx skills are becoming more prevalent due to the inherent ease in programming that it provides. As such, many people can quickly and easily develop Rexx programs to accomplish complex tasks.

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

Considerations

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 used within Rexx programs and the text/binary format used by the underlying callable FTP Client API.

Rexx FTP Client API functions share a common return code format.

3.3.9 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. Once these events are recorded, reports may be generated either by user-written programs that format the data into a readable format or by real time network monitors for display. These records events are referred to as SMF Records, and are stored in one of two places:

- SMF Datasets - VSAM datasets, typically named SYS1.MANx, SYS1.MANy, etc.
- z/OS Dataspaces

SMF datasets must have one dataset 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

These records were either created or enhanced in recent z/OS releases.

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 need to configure ftp.data as well as profile.tcpip 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 3-7:

Example 3-7 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.

Considerations

There are some concerns about the recently enhanced FTP SMF records:

- New fields that have been added to existing FTP SMF records are added at the end of the subtype specific section in each record.
- Users having programs that format SMF data to create reports by using the existing self-defining sections of the record mappings will most likely be unaffected.
- Users who want to use the new fields will have to modify their existing programs.
- Users who have configured Network Management Interface to record SMF records may cause more I/O in the z/OS dataspace.

3.3.10 Recommendations

For users interested in FTP without security, we recommend the configuration discussed in 3.3.1, "FTP without security" on page 125.

For users interested in secure FTP, we recommend either using the configuration discussed in 3.3.2, "FTP with security" on page 125, or using a stack-wide security solution such as AT-TLS or IPSec (discussed in *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342).

For users interested in load balancing across multiple LPARs we recommend load balancing, as discussed in 3.3.5, "Load balancing FTP servers in a sysplex" on page 127.

3.4 How FTP is implemented

Based upon our recommendations from 3.3.10, "Recommendations" on page 131, we show implementation details for the following scenarios:

- FTP without security
- FTP with security

- Load balancing FTP servers in a sysplex
- FTP client via batch
- FTP client via the client API

3.4.1 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.

Implementation tasks

In order to use FTP we need to perform the following tasks.

FTP.DATA for the client

In the LPAR we use for the FTP client, we need to create an FTP.DATA for the client. We copied the sample FTP.DATA for the client from SEZAINST(FTPCDATA) to our PARMLIB. Next, determine if you need to further customize FTPCDATA 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 supplied FTPCDATA 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 recommend that you use a SYSFTPD DD card in your TSO login procedure to ensure that you are using the correct FTP.DATA data set. The FTP client is now ready for use.

Server LPAR

The remaining implementation tasks are performed in the LPAR we use for our FTP server:

1. Define SAF definitions for the FTP server.
2. Create an FTP.DATA for the server.
3. Create a catalogued procedure for the FTP server.
4. Add FTP to the AUTOLOG and PORT statements in PROFILE.TCPIP.
5. Create a port entry in /etc/services.
6. Configure syslogd.
7. Start the FTP server.

SAF definitions

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 may 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 *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

FTP.DATA for the server

First, copy the sample FTP.DATA for the server from SEZAINST(FTPSDATA) to your PARMLIB. 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, and simply uncommented the BANNER and ADMINEMAILADDR statements. The changes we made to the sample FTPSDATA for our typical FTP server installation are shown in Example 3-8.

Example 3-8 Changes to FTPSDATA

```
BANNER /etc/ftpbanner
ADMINEMAILADDRESS user@host.my.net
```

The contents of our /etc/banner file are shown in Example 3-9. Note the /etc/ftpbanner file needs to be readable by the user ID that will start the FTP server. The user ID starting FTP is a root user so file permissions 700 (rwx------) provide sufficient access.

Example 3-9 /etc/banner

```
*****************************************
 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
*****************************************
```

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, the SYSFTPD DD, and the SYSTCPD DD statements need to be updated. Ensure that SYSFTPD refers to the correct FTP.DATA for the server and that the SYSTCPD refers to the resolver configuration data set you want to use. Our procedure is shown in Example 3-10.

Example 3-10 Catalogued procedure for the FTP server

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//*******************************************************************
//*        Descriptive Name:           FTP Server Start Procedure   *
//*        File Name:                  tcpip.SEZAINST(EZAFTPAP)     *
//*                                    tcpip.SEZAINST(FTPD)         *
//*        SMP/E Distribution Name:    EZAFTPAP                     *
//*                                                                 *
//*        Licensed Materials - Property of IBM                     *
//*        "Restricted Materials of IBM"                            *
//*        5694-A01                                                 *
//*        (C) Copyright IBM Corp. 1995, 2005                       *
//*        Status = CSV1R8 *
//*******************************************************************
//*
//* SET PARM1=TCPIVP.TCPPARMS(TCPDATA)
//*
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//*   PARM='/&PARMS'
//*
```

```
//*   Uncomment the SET statement above when using the next two lines.
//*   PARM=('ENVAR("RESOLVER_CONFIG=//''&PARM1''")',
//*   '/&PARMS')
//*
//   PARM=('POSIX(ON) ALL31(ON) ENVAR("_BPX_JOBNAME=FTPDB"',
//   '"TZ=EST")/&PARMS')
//*
//*   PARM=('ENVAR("KRB5_SERVER_KEYTAB=1")',
//*   '/&PARMS')
//*
//**** IVP Note ******************************************************
//*
//* If executing the FTP installation verification procedures (IVP),
//* - Comment the first PARM card and uncomment both lines of the
//*   second PARM card
//* - Uncomment the appropriate SYSFTPD and SYSTCPD DD cards for the IVP
//*
//******************************************************************
//**** _BPX_JOBNAME Note *********************************************
//*
//* The environment variable _BPX_JOBNAME can be specified
//* here in the FTPD procedure, so that all of the logged on
//* FTP users will have the same jobname.  This can then
//* be used for performance control and identifying all FTP users.
//* To use this:
//* - Comment the first PARM card and uncomment both lines of the
//*   third PARM card
//*
//******************************************************************
//**** KRB5_SERVER_KEYTAB Note ***************************************
//*
//* The environment variable KRB5_SERVER_KEYTAB can be specified
//* here in the FTPD procedure, so that the FTP server will use the
//* local instance of the Kerberos security server to decrypt tickets
//* instead of obtaining the key from the key table.
//* To use this:
//* - Comment the first PARM card and uncomment both lines of the
//*   fourth PARM card
//*
//******************************************************************
//*
//*       The C runtime libraries should be in the system's link
//*       list or add them to the STEPLIB definition here.  If you
//*       add them to STEPLIB, they must be APF authorized.
//*
//*       To submit SQL queries to DB2 through FTP, the DB2 load
//*       library with the suffix DSNLOAD should be in the system's
//*       link list, or added to the STEPLIB definition here.  If
//*       you add it to STEPLIB, it must be APF authorized.
//*
//CEEDUMP  DD SYSOUT=*
//*
//*       SYSFTPD is used to specify the FTP.DATA file for the FTP
//*       server.  The file can be any sequential data set, member
//*       of a partitioned data set (PDS), or HFS file.
//*
//*       The SYSFTPD DD statement is optional.  The search order for
//*       FTP.DATA is:
//*
//*           SYSFTPD DD statement
//*          jobname.FTP.DATA
//*          /etc/ftp.data
//*          SYS1.TCPPARMS(FTPDATA)
//*          tcpip.FTP.DATA
//*
//*      If no FTP.DATA file is found, FTP default values are used.
//*      For information on FTP defaults, see z/OS Communications
//*      Server: IP Configuration Reference.
//SYSFTPD DD DISP=SHR,
//         DSN=TCPIPB.TCPPARMS(FTPDB&SYSCLONE)
//*SYSFTPD DD DISP=SHR,
//*        DSN=TCPIVP.TCPPARMS(FTPSDATA)
//*
//*      SYSTCPD explicitly identifies which data set is to be
//*      used to obtain the parameters defined by TCPIP.DATA
//*      when no GLOBALTCPIPDATA statement is configured.
//*      See the IP Configuration Guide for information on
//*      the TCPIP.DATA search order.
//*      The data set can be any sequential data set or a member of
//*      a partitioned data set (PDS).
//SYSTCPD DD DISP=SHR,
//         DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE)
//*SYSTCPD DD DISP=SHR,
//*        DSN=TCPIVP.TCPPARMS(TCPDATA)
//*
```

In our FTPD catalogued procedure, we set BPX_JOBNAME so that all FTP server instances have job name FTPDB. Setting the job name allows syslogd *isolation to function* and allows us to log all messages in a single file, as discussed in Chapter 2, “SYSLOGD” on page 97. We set the TZ environment variable so that messages issued to the console or to SYSLOGD are in the correct time zone, and updated the SYSFTPD and SYSTCPD DD cards.

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 in order to monitor it. The FTP server forks on startup, so the actual job name of the running FTP server may 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 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  FTPDB1 NOAUTOLOG
  21 TCP  FTPDB
```

Port entry in /etc/services

/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 highly recommended that you configure syslogd before starting the FTP server. If you use our recommended syslogd configuration described in Chapter 2, "SYSLOGD" on page 97, then your syslogd configuration file already contains the line 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 *ftp*. Refer to Chapter 2, "SYSLOGD" on page 97, for more details on syslogd.

Example 3-13 Line in /etc/syslog.conf for the FTP server

```
*.FTPDB*.*.* /var/log/%Y/%m/%d/ftp
```

Start the FTP server

Finally, we are ready to start the FTP server using our started procedure.

Verification

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. 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 21:29:05 on 11/27/05.
```

To verify that the FTP server is functional, we used the FTP client and this tested our FTP client as well. Example 3-15 shows the output of our FTP client from another z/OS LPAR after we connected to our FTP server.

Example 3-15 FTP from one z/OS system to another z/OS system

```
IBM FTP CS V1R8
FTP: using TCPIPB
Connecting to:   10.20.1.230 port: 21.
220-FTPDB1 IBM FTP CS V1R8 at WTSC30B.ITSO.IBM.COM, 21:31:05 on 2006-08-25.
220-*****************************************
220- Welcome to the ITSO FTP server
220-
220- This system may be used for management
220- approved purposes only.
220-
220- All transfers are logged.
220-
220- Please report problems to user@host.my.net
220-*****************************************
220 Connection will close if idle for more than 5 minutes.
```

```
NAME (10.20.1.230:CS02):
cs02
>>> USER cs02
331 Send password please.
PASSWORD:

>>> PASS
230 CS02 is logged on.  Working directory is "CS02.".
Command:
```

Example 3-16 shows output from the D J,L command after we logged onto the FTP server. Both the FTP server listener task and our forked task for the connected client are shown. Jobname FTPDB1 running under user ID TCPIP is the FTP server listener. Jobname FTPDB running under user ID CS02 is the FTP server forked task that is serving our connected client.

Example 3-16 D J,L after FTP server is started

```
FTPDB1   STEP1    TCPIP    OWT
FTPDB    STEP1    CS02     OWT  AO
```

Example 3-17 shows the output from a **`netstat sock`** command. Jobname FTPDB1 shows two connections: one on port 21 in a listen state that represents the listener task, and one established connection between local port 21 and 10.20.3.223 port 1031 that represents the established connection for our client (shown in Example 3-15). Note that the established connection for the client shows job name FTPDB1 (the server listener) and not job name FTPDB. This is because the job that accepted the connection was FTPDB and only after the connection was established was a new job forked to handle the FTP session.

Example 3-17 Output of netstat

```
EZD0101I NETSTAT CS V1R8 TCPIPB 419
SOCKETS INTERFACE STATUS:
NAME: FTPDB1    SUBTASK: 007FF540
  TYPE: STREAM  STATUS:  LISTEN    CONN: 0000001E
    BOUNDTO: ::..21
    CONNTO:  ::..0
  TYPE: STREAM  STATUS:  ESTBLSH   CONN: 00000035
    BOUNDTO: ::FFFF:10.20.1.230..21
    CONNTO:  ::FFFF:10.20.3.223..1031
```

Problem determination

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.

3.4.2 FTP with security

We now expand on our base FTP server introduced in 3.4.1, "FTP without security" on page 132, and add additional security options. We set up the secure FTP server on one LPAR, and use a third-party FTP client to demonstrate the secure FTP session

Implementation tasks

In order to start FTP with TLS we need to first perform all the implementation tasks for both client and server described in 3.4.1, "FTP without security" on page 132. Next we need to create a server certificate in the server LPAR. Then we make changes to the FTP.DATA file for the server. Finally, we need to **permit** the user ID of the FTP server to the appropriate RACF classes.

Certificate generation

We strongly encourage using a digital certificate that is signed by a professional certificate authority (CA). However, to demonstrate the security features of the FTP we use self-signed certificates that we create with the following RACF commands:

1. First, we create a server key ring:

   ```
   RACDCERT ID(TCPIP) ADDRING(SERVERRING)
   ```

2. Then we generate a RACF server certificate:

   ```
   RACDCERT ID(TCPIP)  GENCERT  SUBJECTSDN(CN('UNIT1') OU('TESTING') C('US') ) TRUST
   WITHLABEL('FTP_SERVER') SIZE(1024)
   ```

3. Finally, we add our newly generated certificate to our newly created server key ring:

   ```
   RACDCERT ID(TCPIP) CONNECT(ID(TCPIP) LABEL('FTP_SERVER') RING(SERVERRING) DEFAULT)
   ```

Update the FTP.DATA for the server

We added a number of statements to the server FTP.DATA to provide for additional security:

1. Add a PORTCOMMAND REJECT statement to prevent the use of our FTP server in three-way FTP transfers.
2. Enable TLS with a EXTENSIONS AUTH_TLS statement.
3. Add a KEYRING statement to identify the location of our keyring file.
4. Add a CIPHERSUITE statement to allow for a number of authentication and encryption algorithms.
5. Add a SECURE_FTP ALLOWED statement to allow for, but not require, TLS clients.
6. Add a SECURE_DATACONN PRIVATE statement to encrypt FTP data connection.

Example 3-18 shows the additional statements we added to our FTP.DATA data set to enable security.

Example 3-18 Additional statements in FTP.DATA to enable security

```
PORTCOMMAND REJECT
EXTENSIONS AUTH_TLS
KEYRING SERVERRING
CIPHERSUITE SSL_RC4_MD5
SECURE_FTP ALLOWED
SECURE_DATACONN PRIVATE
```

Update RACF

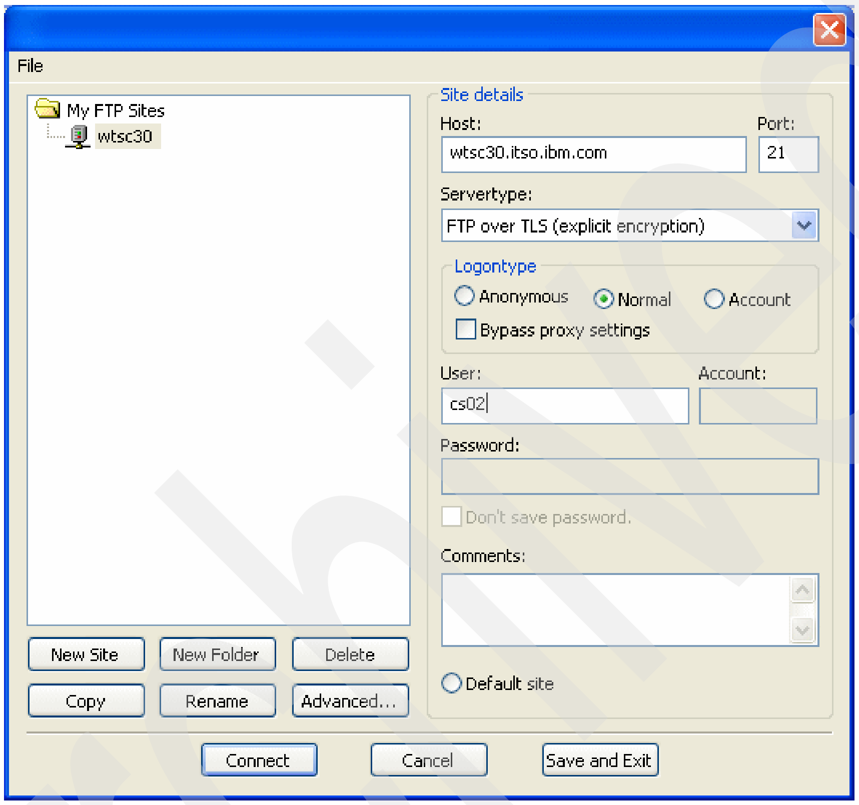
When we first tried to use FTP with TLS security the TLS negotiation failed. In reviewing the FTP server trace we found a message indicating that the TLS initialization had failed because permission was denied. Upon reviewing the console log, we found that RACF READ access was required for the IRR.DIGTCERT.LISTRING resource in the FACILITY class. Once we granted READ access to the FTP server user ID we were able to perform TLS negotiation. To grant READ access, we issued these commands:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(TCPIP) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Verification

We verified that FTP server was up and running using the same verification steps as in 3.4.1, “FTP without security” on page 132. However, in order to verify our FTP server is really using TLS, we needed an FTP client that supports TLS. We chose to use Filezilla, an open source TLS-enabled FTP client that is available at: `http://filezilla.sourceforge.net`

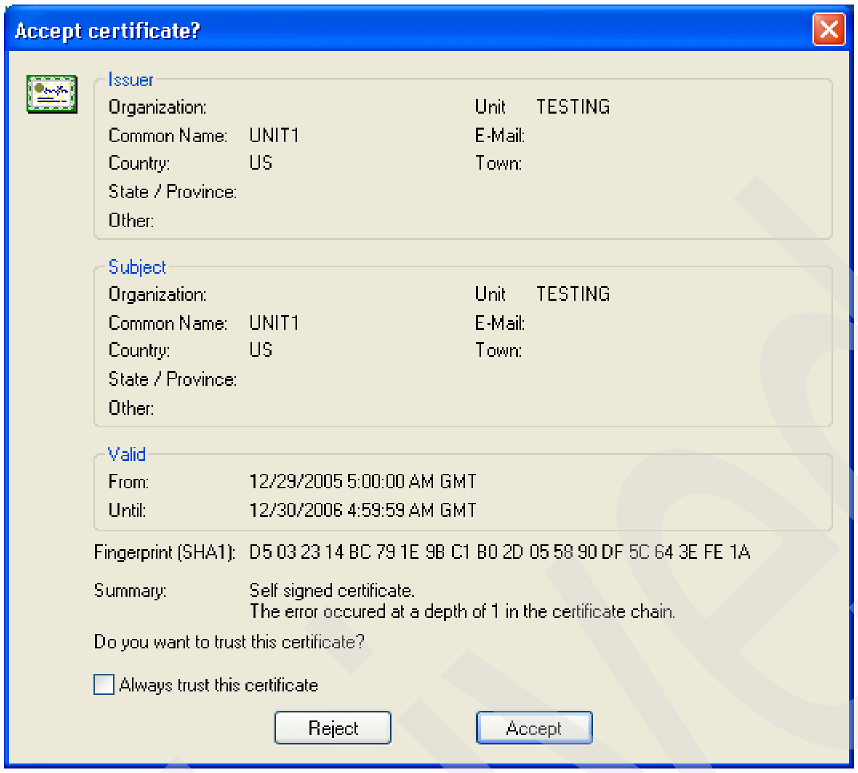
The following figures show the configuration of Filezilla. First, in Figure 3-4 we show the configuration settings needed to connect to our TLS-enabled secure FTP server.



Host: wtsc30.itso.ibm.com
Port: 21
Servertype: FTP over TLS (explicit encryption)
Logontype: Anonymous [✓ Normal] Account
Bypass proxy settings
User: cs02
Account:
Password:
Don't save password.
Comments:
Default site

Figure 3-4 Configuration of Filezilla to connect to our TLS-enabled secure FTP server

Next, in Figure 3-5 we show Filezilla confirming that we want to accept the self-signed certificate.



Issuer
Organization: | Unit TESTING
Common Name: UNIT1 | E-Mail:
Country: US | Town:
State / Province:
Other:

Subject
Organization: | Unit TESTING
Common Name: UNIT1 | E-Mail:
Country: US | Town:
State / Province:
Other:

Valid
From: 12/29/2005 5:00:00 AM GMT
Until: 12/30/2006 4:59:59 AM GMT

Fingerprint (SHA1): D5 03 23 14 BC 79 1E 9B C1 B0 2D 05 58 90 DF 5C 64 3E FE 1A

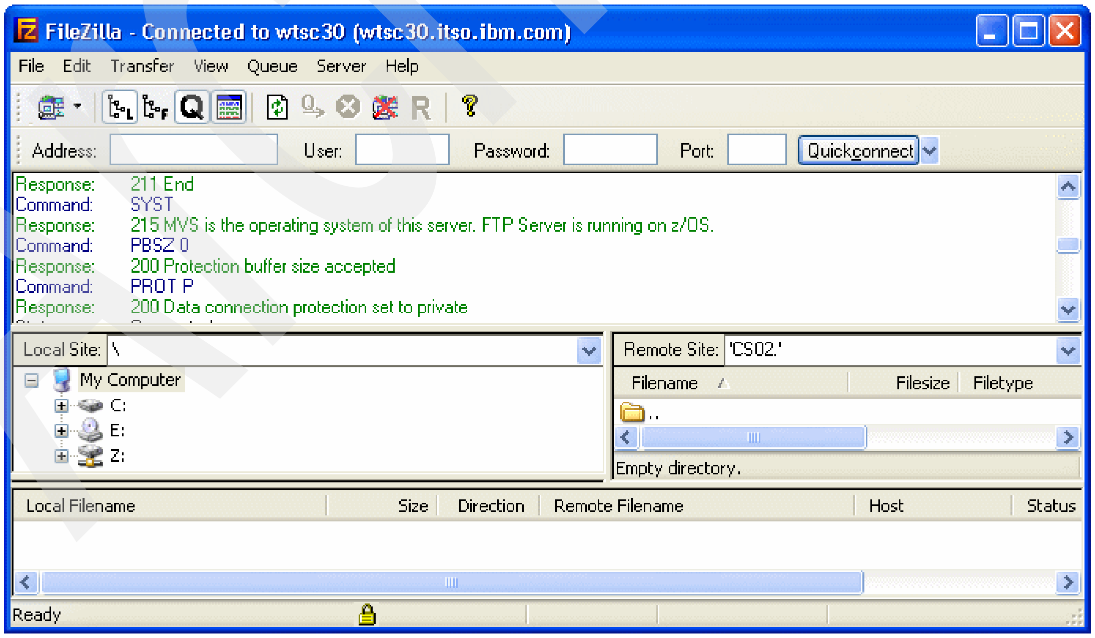
Summary: Self signed certificate.
The error occured at a depth of 1 in the certificate chain.

Do you want to trust this certificate?

☐ Always trust this certificate

Figure 3-5 Filezilla confirming that the self-signed certificate is acceptable

Finally, in Figure 3-6 we have confirmation that Filezilla was able to connect to our secure FTP server. Note the lock icon in the lower center of the screen as well as the messages indicating that the data connection protection was set to private.



```
Response:   211 End
Command:    SYST
Response:   215 MVS is the operating system of this server. FTP Server is running on z/OS.
Command:    PBSZ 0
Response:   200 Protection buffer size accepted
Command:    PROT P
Response:   200 Data connection protection set to private
```

Local Site: \
My Computer
C:
E:
Z:

Remote Site: 'CS02.'
Filename | Filesize | Filetype
..
Empty directory.

Local Filename | Size | Direction | Remote Filename | Host | Status

Ready

Figure 3-6 Confirmation that Filezilla connected to our system using TLS

Problem determination

Refer to the problem determination aids discussed in 3.4.1, “FTP without security” on page 132.

3.4.3 Load balancing FTP servers in a sysplex

Multiple FTP servers can accept connections that are distributed by the SD. In our environment we set up a sysplex using two stacks, both with the job name of TCPIPB that is in two LPARS, SC30 and SC31. The same sysplex environment is discussed in more detail in 1.3.4, “Multiple Telnet servers with SD” on page 17.

Implementation tasks

In order to load balance FTP in a sysplex we need to first perform all the implementation tasks for both client and server in 3.4.1, “FTP without security” on page 132, or, if security is desired, perform all the tasks in 3.4.2, “FTP with security” on page 137. Next, we need to perform these additional tasks:

1. Customize a second TCP/IP stack started task procedure.
2. Customize both TCP/IP stack configuration profile data sets for sysplex.
3. Customize a second FTP server started task procedure.
4. Customize a second FTP server configuration profile data set.
5. Start the stacks.
6. Start the FTP servers.

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 CS30 and our second TCP/IP task is also named TCPIPB and is running in LPAR SC31. 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 configuration profile data sets for sysplex

The stack in LPAR SC30 is our SD stack, and the stack in LPAR SC31 is the backup stack. Both stacks are FTP targets. For a complete discussion and examples of setting up a stack, refer to:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

We need to 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 SC30 are shown in Example 3-19.

Example 3-19 Sysplex enablement for TCPIPB on SC30

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.100 255.255.255.0 8
;
VIPADYNAMIC
```

```
  ;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.20.10.1    PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using ROUNDROBIN         -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.21  ;FTP General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.20.10.21   PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM            -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.22  ;FTP Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.20.10.22   PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using SERVERWLM          -
  ;---------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.23  ;FTP Payrol
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.20.10.23   PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101
  ;---------------------------------------------------------------------
  ;      Distribute to 10.20.20.100 via normal XCF   (no viparoute)  -
  ;      Distribute to 10.20.20.101 via IP routing   (   viparoute)  -
  ;---------------------------------------------------------------------
 ;;VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
   VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa

 ENDVIPADYNAMIC
```

The statements necessary to enable sysplex distribution in TCPIPB on LPAR SC31 are shown in Example 3-20.

Example 3-20 Sysplex enablement for TCPIP on SC31

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.101 255.255.255.0 8
;
VIPADYNAMIC
  ;---------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.20.10.1    PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using ROUNDROBIN         -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.21  ;FTP General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.20.10.21   PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM            -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.22  ;FTP Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.20.10.22   PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using SERVERWLM          -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.23  ;FTP Payrol
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.20.10.23   PORT 21
                  DESTIP 10.20.20.100
                         10.20.20.101

  ;-----------------------------------------------------------------
  ;      Distribute to 10.20.20.100 via IP routing   (   viparoute)  -
  ;      Distribute to 10.20.20.101 via normal XCF   (no viparoute)  -
  ;-----------------------------------------------------------------
   VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa

 ENDVIPADYNAMIC
```

Customize a second FTP server started task procedure

This second started task is for our second FTP server on TCPIPB in LPAR SC31. It 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 3-10 on page 133.

Customize a second FTP server configuration profile data set

The FTP.DATA for both servers should be identical. The FTP.DATA file we used for both FTP servers is discussed in 3.3.1, “FTP without security” on page 125, under the heading “FTP.DATA for the server” on page 133.

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 73.

Customize an OMPROUTE configuration to support 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 in order to give them their uniqueness: interface IP addresses and router IDs are examples. For details on configuring OMPROUTE, see:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *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 to support the second OMPROUTE" on page 73.

Start the stacks

Issue the MVS START command on both systems:

```
S TCPIPB
```

Start the FTP servers

If FTP is autologged in both starts, 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
```

Verification for FTP servers with SD

All of the verification tasks previously discussed in 3.3.1, "FTP without security" on page 125, and 3.3.2, "FTP with security" on page 125, 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 on the NETSTAT command, refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

The following additional tasks to verify sysplex distribution to FTP servers are discussed below:

- 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 3-21 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 3-21 NETSTAT VCRT on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VCRT
   EZD0101I NETSTAT CS V1R8 TCPIPB 282
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.20.10.21..21
     SOURCE:  10.20.1.241..1029
     DESTXCF: 10.20.20.100
   DEST:      10.20.10.21..21
     SOURCE:  10.20.2.222..1026
     DESTXCF: 10.20.20.101
   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 seen in Example 3-22.

Example 3-22 NETSTAT VCRT on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VCRT
   EZD0101I NETSTAT CS V1R8 TCPIPB 664
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.20.10.21..21
     SOURCE:  10.20.2.222..1026
     DESTXCF: 10.20.20.101
   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 3-23 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: Notice that 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 3-23 NETSTAT VDPT on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VDPT
   EZD0101I NETSTAT CS V1R8 TCPIPB 315
   DYNAMIC VIPA DESTINATION PORT TABLE:
   DEST:        10.20.10.1..21
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.1..21
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.21..21
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000002  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.20.10.21..21
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000001  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.20.10.22..21
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.22..21
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.20.10.23..21
     DESTXCF:   10.20.20.100
     TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
     FLG: SERVERWLM
   DEST:        10.20.10.23..21
     DESTXCF:   10.20.20.101
     TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
     FLG: SERVERWLM
   10 OF 10 RECORDS DISPLAYED
   END OF THE REPORT
```

SC31 is not a distributor at the moment; therefore, it shows no information, as seen in Example 3-24.

Example 3-24 NETSTAT VDPT on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VDPT
   EZD0101I NETSTAT CS V1R8 TCPIPB 666
   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. The primary distributor shows the VIPA DEFINE, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 3-25.

Example 3-25 NETSTAT VIPADCFG on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADCFG
   EZD0101I NETSTAT CS V1R8 TCPIPB 351
   DYNAMIC VIPA INFORMATION:
     VIPA DEFINE:
       IPADDR/PREFIXLEN: 10.20.10.1/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.20.10.21/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.20.10.22/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.20.10.23/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
```

```
    VIPA RANGE:
      IPADDR/PREFIXLEN: 10.20.10.240/28
        MOVEABLE: NONDISR
    VIPA DISTRIBUTE:
      DEST:        10.20.10.1..21
        DESTXCF:   10.20.20.100
          SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
      DEST:        10.20.10.1..21
        DESTXCF:   10.20.20.101
          SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
      DEST:        10.20.10.21..21
        DESTXCF:   10.20.20.100
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.20.10.21..21
        DESTXCF:   10.20.20.101
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.20.10.22..21
        DESTXCF:   10.20.20.100
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.20.10.22..21
        DESTXCF:   10.20.20.101
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.20.10.23..21
        DESTXCF:   10.20.20.100
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      DEST:        10.20.10.23..21
        DESTXCF:   10.20.20.101
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    VIPA ROUTE:
      DESTXCF:     10.20.20.101
        TARGETIP:  10.20.1.241
END OF THE REPORT
```

The backup stack shows the VIPA BACKUP, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 3-26.

Example 3-26 NETSTAT VIPADCFG on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADCFG
   EZD0101I NETSTAT CS V1R8 TCPIPB 668
   DYNAMIC VIPA INFORMATION:
     VIPA BACKUP:
       IPADDR/PREFIXLEN: 10.20.10.1
         RANK: 200  MOVEABLE:           SRVMGR:
       IPADDR/PREFIXLEN: 10.20.10.21
         RANK: 200  MOVEABLE:           SRVMGR:
       IPADDR/PREFIXLEN: 10.20.10.22
         RANK: 200  MOVEABLE:           SRVMGR:
       IPADDR/PREFIXLEN: 10.20.10.23
         RANK: 200  MOVEABLE:           SRVMGR:
     VIPA RANGE:
       IPADDR/PREFIXLEN: 10.20.10.240/28
         MOVEABLE: NONDISR
     VIPA DISTRIBUTE:
       DEST:        10.20.10.1..21
         DESTXCF:   10.20.20.100
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.20.10.1..21
         DESTXCF:   10.20.20.101
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
```

```
      DEST:        10.20.10.21..21
        DESTXCF:   10.20.20.100
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.20.10.21..21
        DESTXCF:   10.20.20.101
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.20.10.22..21
        DESTXCF:   10.20.20.100
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.20.10.22..21
        DESTXCF:   10.20.20.101
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.20.10.23..21
        DESTXCF:   10.20.20.100
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      DEST:        10.20.10.23..21
        DESTXCF:   10.20.20.101
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    VIPA ROUTE:
      DESTXCF:     10.20.20.100
        TARGETIP:  10.20.1.230
   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 3-27 shows SC30 information.

Example 3-27 NETSTAT VIPADYN on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN
   EZD0101I NETSTAT CS V1R8 TCPIPB 365
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.23/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
   VIPA ROUTE:
     DESTXCF: 10.20.20.101
       TARGETIP: 10.20.1.241
       RTSTATUS: ACTIVE
   5 OF 5 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 3-28 shows the same information for SC30.

Example 3-28 NETSTAT VIPADYN,DVIPA on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
   EZD0101I NETSTAT CS V1R8 TCPIPB 378
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.20.10.23/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
   4 OF 4 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 3-29 shows SC30 Netstat output for VIPADYN,VIPAROUTE, with filters for viparoute only.

Example 3-29   NETSTAT VIPADYN,VIPAROUTE on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
   EZD0101I NETSTAT CS V1R8 TCPIPB 387
   VIPA ROUTE:
     DESTXCF: 10.20.20.101
       TARGETIP: 10.20.1.241
       RTSTATUS: ACTIVE
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 3-30 shows SC31Netstat output for VIPADYN ftp and FTP dvipa addresses.

Example 3-30   NETSTAT VIPADYN on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN
   EZD0101I NETSTAT CS V1R8 TCPIPB 670
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.23/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
   VIPA ROUTE:
     DESTXCF: 10.20.20.100
       TARGETIP: 10.20.1.230
       RTSTATUS: ACTIVE
   5 OF 5 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 3-31 shows SC31 Netstat output for VIPADYN,DVIPA: with filters for dvipa only.

Example 3-31   NETSTAT VIPADYN,DVIPA on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
   EZD0101I NETSTAT CS V1R8 TCPIPB 672
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.20.10.1/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.21/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
     IPADDR/PREFIXLEN: 10.20.10.22/24
       STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
```

```
    IPADDR/PREFIXLEN: 10.20.10.23/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
  4 OF 4 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 3-32 shows SC31 Netstat output for VIPADYN,VIPAROUTE, with filters for viparoute only.

Example 3-32 NETSTAT VIPADYN,VIPAROUTE on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
   EZD0101I NETSTAT CS V1R8 TCPIPB 674
   VIPA ROUTE:
     DESTXCF: 10.20.20.100
       TARGETIP: 10.20.1.230
       RTSTATUS: ACTIVE
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

Problem determination for FTP servers with SD

Refer to the problem determination steps discussed in 3.3.1, “FTP without security” on page 125, and 3.3.2, “FTP with security” on page 125. Also refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

4

Simple Network Management Protocol

Simple Network Management Protocol (SNMP) enables monitoring and management of the devices and computers participating in an TCP/IP network. This chapter focuses on the SNMP services that are available in the z/OS V1R8 Communications Server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

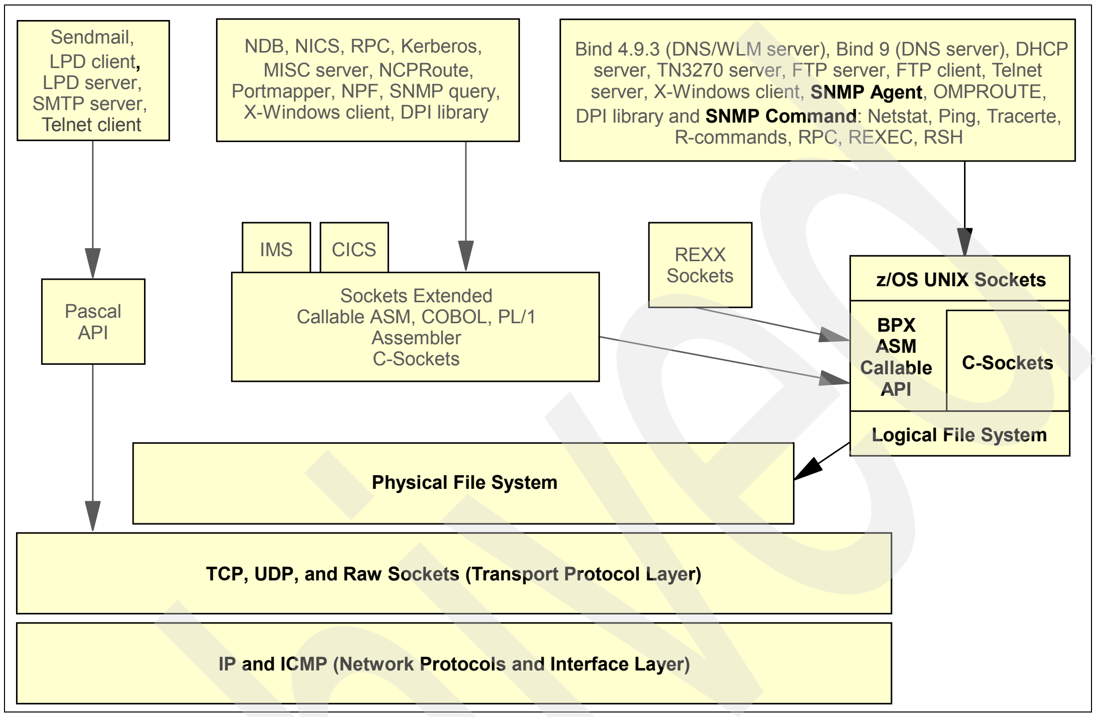
The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains comprehensive descriptions of the individual parameters for setting up SNMP services on z/OS. It also includes step-by-step checklists and some supporting examples. It is not the intent of this book to duplicate the information in the manual, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 4.1.2, “Additional information sources for SNMP” on page 155.

This chapter discusses the following.

Section	Topic
4.1, “Overview” on page 152	Discusses the basic concepts of SNMP network management
4.2, “Why SNMP is important” on page 155	Discusses key characteristics of SNMP and why network management is important in your environment
4.3, “The common design scenarios for SNMP” on page 156	Presents commonly implemented SNMP design scenarios, their dependencies, advantages, considerations, and our recommendations
4.4, “How the z/OS SNMP support is implemented” on page 157	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

4.1 Overview

As illustrated in Figure 4-1, SNMP is one of the standard applications provided with the z/OS Communications Server.



Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270 server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
REXX Sockets
z/OS UNIX Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

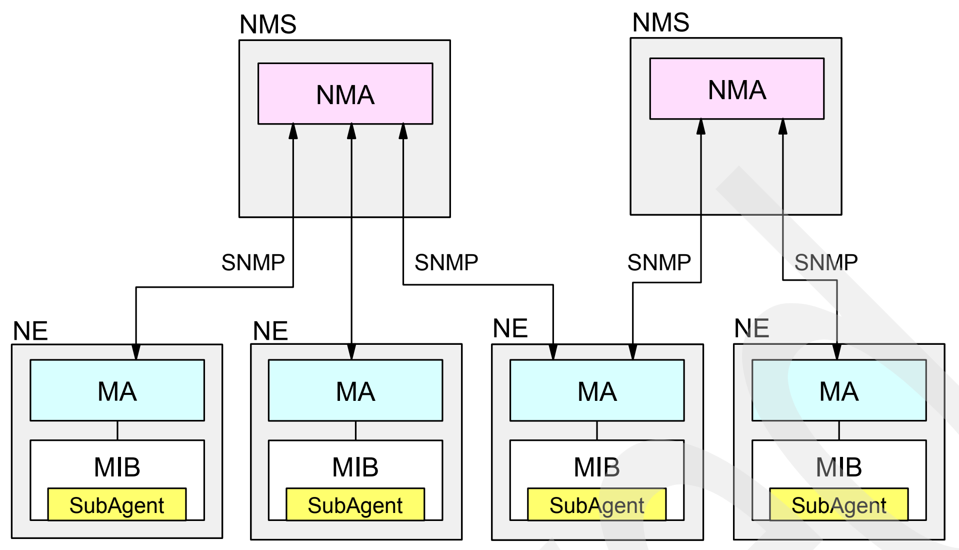
Figure 4-1 z/OS SNMP management services

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. SNMP is an Internet standard protocol. Its current specification can be found in RFC 1157 Simple Network Management Protocol (SNMP).

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.

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 diagram in Figure 4-2 on page 153 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 4-2   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.

Two SNMP implementations are supported in z/OS:

- Simple Network Management Protocol (SNMP) agent and subagent in the MVS environment
- The z/OS UNIX snmp command as a manager requesting information from the agent

4.1.1 Basic concepts of SNMP

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 4-3 on page 154.

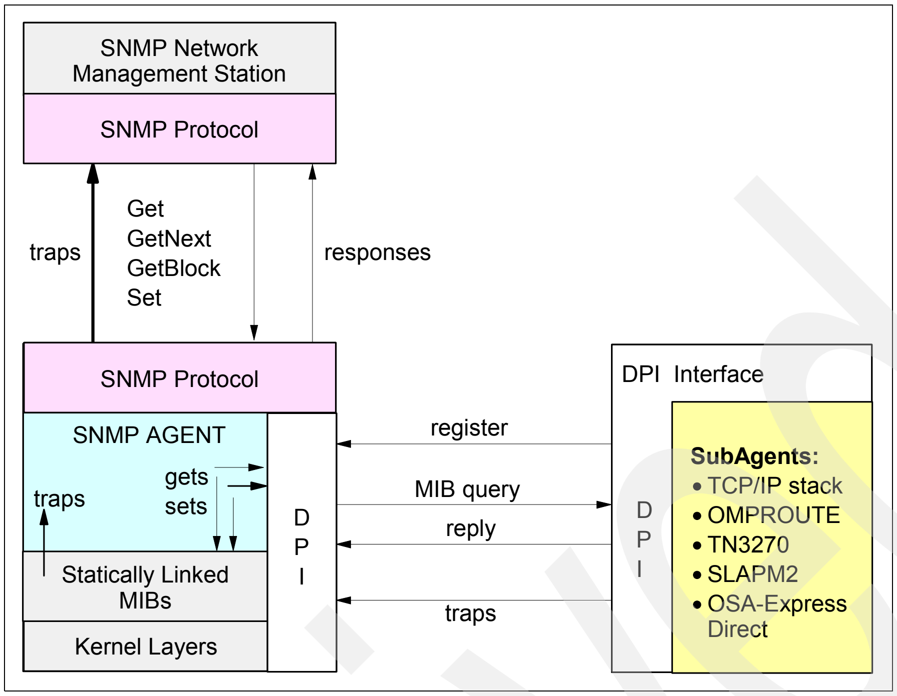


Figure 4-3 SNMP agent/subagent relationship to a Network Management Station (manager)

The SNMP Distributed Programming Interface (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 via 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.
- Once 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 on using the DPI interface see *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787.

Some of the subagents and their specific MIBs delivered with the z/OS Communications Server are:

- 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 via 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 on the standards and how they relate to each other, refer to *TCP/IP Tutorial and Technical Overview*, GG24-3376.

4.1.2 Additional information sources for SNMP

Detailed information for the SNMP protocols can be found in the following documents:

- *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.

4.2 Why SNMP is important

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.

4.3 The common design scenarios for SNMP

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, and to set up the z/OS UNIX `snmp` command as an SNMP manager.

4.3.1 z/OS SNMP agent

We discuss the configuration of OSNMPD as an agent in the following topics:

- Description of the z/OS SNMP agent
- Dependencies of the z/OS SNMP agent
- Advantages of using the z/OS SNMP agent
- Considerations for using the z/OS SNMP agent

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 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.

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 up 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. 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, 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.

Advantages of using the z/OS SNMP agent

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 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.

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, refer to the SNMP chapter in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

4.3.2 z/OS SNMP client command

The two SNMP client applications provided with the z/OS Communications Server are:

- SNMP command from the NetView environment
- **snmp** command in the z/OS shell

The SNMP command in the NetView environment requires the use of the NetView product. It supports SNMP version 1. The **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 below. For complete details, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

4.3.3 Recommendations for using SNMP

Depending on the network management requirements of your organization, you may have to implement certain features of the SNMP agent on your z/OS platform. We show you how to set up the basic SNMP agent support and how to activate subagent support in a few other z/OS application products. In order to use SNMP, you need to consider:

- The z/OS SNMP agent
- The z/OS SNMP subagents
- The z/OS UNIX **snmp/osnmp** command

4.4 How the z/OS SNMP support is implemented

The following discussions describe our implementations of:

- z/OS SNMP agent setup and usage
- z/OS SNMP client (command) setup and usage

4.4.1 z/OS SNMP agent setup and usage

There are common steps to set up the SNMP agent regardless of how it is going to be used or what role it plays. The z/OS SNMP agent and its relationship with the IP network is illustrated in Figure 4-4.

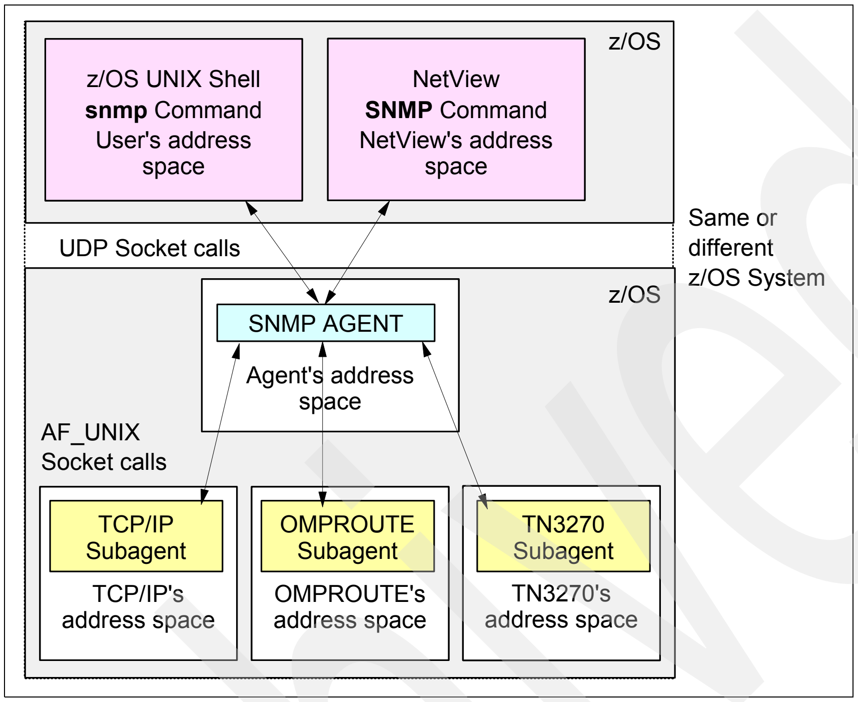


Figure 4-4 SNMP agent/subagent relationships

The subagents are set up independently. Each has its own set of definitions and configuration file. The diagram in Figure 4-4 is referred to and explained in the following sections:

- Implementation tasks for the z/OS SNMP agent
- Configuration examples for the z/OS SNMP agent
- Implementation of the z/OS Communications Server subagents
- Configuration examples of the z/OS Communications Server subagents
- Verification of the z/OS SNMP agent
- Problem determination tasks for the z/OS SNMP agent

Implementation tasks for the z/OS SNMP agent

The steps to configure the z/OS SNMP server are discussed here. Examples of each step are listed in "Configuration examples for the z/OS SNMP agent" on page 162.

1. Update the TCPIP profile configuration data set for the agent.
2. Update the TCPIP profile configuration data set for the query engine.
3. Update the security server, such as RACF, to define the SNMP started task.
4. Customize the SNMP procedure JCL.
5. Create the SNMP agent MIB object configuration information.
6. Determine the type of security the agent should support.
7. Configure the query engine started task (optional).
8. Configure the trap forwarder started task (optional).
9. Start the SNMP-related started tasks.

Update the TCPIP 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. This is 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 TCPIP 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 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 the security server, such as RACF, to define the SNMP 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:

- OSNMPD, the agent's started task
- SNMPQE, the query engine
- TRAPFWD, the trap forwarder

The OSNMP agent started task is used as an example. This discussion assumes RACF is the security subsystem being used. If another security product is used, refer to 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 OSNMP*.* STDATA(USER(OSNMP))
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 OSNMP
ALTUSER OSNMP OMVS(UID(0) PROGRAM ('/bin/sh') HOME('/'))
```

In this example the user ID name is OSNMP, 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 OSNMP OMVS(UID(25) PROGRAM ('/bin/sh') HOME('/'))
      PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(OSNMP) ACCESS(READ)
      ```

      In this example, the user ID is OSNMP 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 procedure JCL

A sample of the procedure is in hlq.SEZAINST(OSNMPDPR). Customize data set names to meet installation standards. There are some follow-on steps below that may 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 TCPIP profile configuration data set.

Create the SNMP agent MIB object configuration information

Supplying this information permits you to customize the values of certain MIB object 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.

Determine the type of security the agent should support

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. Refer to *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 setup 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 now 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 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.

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 refer to Example 4-8 on page 163 for an example SNMPQE Proc JCL.

Configure the trap forwarder started task (optional)

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. Refer to:

- Example 4-10 on page 164 showing the Trap Forwarder Procedure JCL
- Example 4-11 on page 164 showing the Standard environment variables file
- Example 4-12 on page 164 showing the forwarder's configuration file

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.

Start the SNMP-related started tasks

If the SNMP agent encounters any errors processing its configuration files, error messages are written to syslogd, not to the console.

```
S SNMPDB
S SNMPQEB
S TRAPFWD
```

Configuration examples for the z/OS SNMP agent

Our SNMP agent procedure is shown in Example 4-3.

Example 4-3 SNMP agent Proc JCL

```
//SNMPDB PROC PARMS='-d 0'
//SNMPDB EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/&PARMS')
//STDENV  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(SENVB&SYSCLONE.)
//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=*
```

Portions of our SNMP agent configuration data sets are presented in:

- Example 4-4 shows STDENV files.
- Example 4-5 shows the community name file pw.src.
- Example 4-6 on page 163 shows the snmp trapdest file.
- Example 4-7 on page 163 shows the osnmpd_data file.
- Example 4-8 on page 163 shows the SNMPQE proc for the query engine.
- Example 4-9 on page 163 shows NetView SNMPARMS for use with query engine.
- Example 4-10 on page 164 shows the TRAPFWD proc of the forwarder.
- Example 4-11 on page 164 shows the STDENV files for TRAPFWD on SC30.
- Example 4-12 on page 164 shows the Trap Forwarder Configuration files for each.

Example 4-4 SNMP agent STDENV files for SC30 and SC31

```
for SC30:
BROWSE    TCPIPB.TCPPARMS(SENVB30) - 01.01          Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB30)'
OSNMPD_DATA=//'TCPIPB.TCPPARMS(SNMPDB30)'
PW_SRC=//'TCPIPB.TCPPARMS(PWSRCB)'
SNMPTRAP_DEST=//'TCPIPB.TCPPARMS(STRPDSTB)'

for SC31:
BROWSE    TCPIPB.TCPPARMS(SENVB31) - 01.01          Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
OSNMPD_DATA=//'TCPIPB.TCPPARMS(SNMPDB31)'
PW_SRC=//'TCPIPB.TCPPARMS(PWSRCB)'
SNMPTRAP_DEST=//'TCPIPB.TCPPARMS(STRPDSTB)'
```

Note: When using _CEE_ENVFILE with an MVS data set, the data set must be allocated with RECFM=V. RECFM=F is not recommended, because RECFM=F enables padding with blanks for the value of environment variables. A file system name could result in including all the blanks after its name designation, causing *file not found* conditions in the UNIX environment.

Example 4-5 SNMP PW.SRC file

```
BROWSE    TCPIPB.TCPPARMS(PWSRCB) - 01.00           Line 00000000 Col 001 080
```

```
public  0.0.0.0   0.0.0.0
NSS 9.15.97.0       255.255.255.0
nss 9.15.97.0       255.255.255.0
ral 9.20.0.0         255.255.0.0
RAL 9.20.0.0         255.255.0.0
jOs9m2ap 10.0.0.0       255.0.0.0
JOS9M2AP 10.0.0.0       255.0.0.0
```

Example 4-6 SNMPTRAP.DEST file

```
BROWSE    TCPIPB.TCPPARMS(STRPDSTB) - 01.00          Line 00000000 Col 001 080
# IP ADDRESSES OF THE SYSTEMS WHERE TRAPS CAN BE FORWARDED TO
10.20.1.230 UDP
10.20.1.241 UDP
10.20.1.221 UDP
```

Example 4-7 OSNMPD_DATA file TCPIPB.TCPPARMS(SNMPDB30)

```
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "I&O Mainframe Network Support"
  sysLocation "Datacenter Mainframe Operations"
  sysName "SC30 - z/OS V1R8 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 Communication 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    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
```

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
```

Example 4-9 SNMPARMS NetView member interface to SNMPQE

```
  SNMPQE   SNMPQEB  * Userid of SNMP Query Engine
  SNMPQERT 60       * Retry timer (seconds) for IUCV CONNECT
  SNMPRCNT 2        * Retry count for sending SNMP requests
  SNMPRITO 10       * Retry initial timeout (10ths of a second)
  SNMPRETO 2        * Retry backoff exponent (1=linear, 2=exponential)
```

```
SNMPMMLL 80       * Line length for Multiline Messages 38/44
```

Note: If you are going to run both the SNMP Query Engine started task and the trap forwarder started task, then trap forwarder will have to listen on a different port from 162 to avoid conflicts. An alternative would be to let them both listen on port 162, but cause the trap forwarder to be a bind-specific listener by coding BIND on the port reservation statement and specifying a different address for the forwarder. Client threads sending packets to the forwarder would then use that address to contact the forwarder.

Example 4-10 Trap forwarder proc JCL, TRAPFWDB

```
BROWSE    TCPIPB.TCPPARMS(TRAPFWDB) - 01.01          Line 00000000 Col 001 080
//TRAPFWDB PROC PARMS='-d 0',TRAPENV=TENVB&SYSCLONE.
//TRAPFWDB EXEC PGM=EZASNTRA,REGION=0M,TIME=NOLIMIT,
//         PARM=('POSIX(ON) ALL31(ON)',
//         'ENVAR("_CEE_ENVFILE=DD:STDENV")/-p 1162 &PARMS')
//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=*
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TRAPENV.)
```

Example 4-11 STDENV file for the TRAPFWDB task on system SC30

```
BROWSE    TCPIPB.TCPPARMS(TENVB30) - 01.02          Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB30)'
TRAPFWD_CONF=//'TCPIPB.TCPPARMS(TCFGB30)'
```

Example 4-12 TRAPFWD.CONF files for the TRAPFWDB task on each system

```
BROWSE    TCPIPB.TCPPARMS(TCFGB30) - 01.00          Line 00000000 Col 001 080
 10.20.1.241  161
 10.20.1.221  161

BROWSE    TCPIPB.TCPPARMS(TCFGB31) - 01.01           Line 00000000 Col 001 080
 10.20.1.230  161
 10.20.1.221  161

BROWSE    TCPIPB.TCPPARMS(TCFGB32) - 01.01           Line 00000000 Col 001 080
 10.20.1.230  161
 10.20.1.241  161
```

Implementation of the z/OS Communications Server 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 next.

TCP/IP subagent

The TCP/IP subagent in the z/OS Communications Server is a z/OS UNIX application that runs in its own task 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. In order 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 4-25 on page 174 and Example 4-26 on page 175.
- 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 to 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 security server requirements. See the security discussion earlier in this book in “Update the security server, such as RACF, to define the SNMP started task” on page 159. For details on configuring this subagent, refer to *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

Note: The SLA subagent (pagtsnmp) became obsolete in z/OS V1R8; therefore, you should use the Network SLAPM2 (nslapm2) subagent instead of it.

OSA-Express Direct subagent

The OSA product also provides an SNMP subagent that supports management data for OSA-Express features, called he 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, as well as standard RFCs. The data is supported by the SNMP TCP/IP subagent. This subagent runs as its own started task and has security server requirements. See the security discussion earlier in this book in “Update the security server, such as RACF, to define the SNMP started task” on page 159.

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 server’s MCL. The file can be acquired and installed by using one of these two methods:

- Selecting Advanced Functions on the Support Element
- Downloading from:
  `https://www-1.ibm.com/servers/resourcelink/lib03010.nsf/0/A310C9113AA36A6485256BBA00697BDA?OpenDocument`
  a. After logging in, select **Library**.
  b. Under Library shortcuts on the right side of the screen, select **Open System Adapter (OSA) Library**.
  c. Select **OSA-Express SNMP Direct MIB Module** for a description, or click **TXT** for the module.
  d. Save the MIB file to the location required by your SNMP management application.

Note: If you subscribe to the document “OSA-Express Direct SNMP MIB module” through Resource Link™, then you will receive e-mail notification of document changes.

Configuration examples of the z/OS Communications Server subagents

Example 4-13 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 4-13 Enabling subagents within their respective configuration profiles

```
TCP/IP stack subagent:
   SACONFIG ENABLED AGENT 161 COMMUNITY j0s9m2ap

OMPROUTE subagent:
   ROUTESA_CONFIG ENABLED=YES AGENT=161 COMMUNITY=”j0s9m2ap”;

TN3270E Telnet server subagent:
   TNSACONFIG ENABLED AGENT 161 COMMUNITY j0s9m2ap
```

An example of the NSLAPM2 procedure JCL is shown in Example 4-14. Our started task was called SLAPM2B.

Example 4-14 NSLAPM2 Proc JCL, SLAPM2B, subagent setup information about PARM statement

```
BROWSE    SYS1.PROCLIB(SLAPM2B) - 01.02             Line 00000000 Col 001 080
//SLAPM2B PROC STDENV=NENVB&SYSCLONE.
//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')
//STDENV   DD DSN=TCPIPB.TCPPARMS(&STDENV.),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Typical startup messages for NSLAPM2 are shown in Example 4-15.

Example 4-15 SLAPM2B startup messages

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 0  --  N O D E

23.57.16 STC08401 ---- MONDAY,    24 OCT 2006 ----
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
```

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 4-16. Our started task was called SNMPOSAB.

Example 4-16 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=*
```

Verification of the z/OS SNMP agent

Some of the verification actions you can perform 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. Samples of the initialization messages are shown in Example 4-17, and samples of the termination messages are shown in Example 4-18 on page 168.

Example 4-17 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 4-18 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 10/21/2006 16:45:15 SNMP RC -5. Disconnecting from Agent
   IOB002I 10/21/2006 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 4-19. 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 once the SNMP task is restarted.

Example 4-19 IOBSNMP initialization and termination messages

```
S SNMPOSAB
IOB000I 10/21/2006 16:40:07 Starting OSA SNMP subagent
Version 1.2.36, June  24, 2006, APAR PK02886
IOB028I 10/21/2006 16:40:07 Using stack name TCPIPB
IOB021I 10/21/2006 16:40:07 OSA SNMP subagent initialization complete

P SNMPOSB
IOB031E 10/21/2006 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 4-20.

Example 4-20 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

Refer to 4.4.2, "z/OS SNMP client (command) setup and usage" on page 170 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

Refer to 4.4.2, "z/OS SNMP client (command) setup and usage" on page 170 for information about how to use the **snmp** command. By changing particular MIB values, a TCP/IP stack configuration can be altered without recycling. Some of the operations supported are:

- 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

Refer to 4.4.2, "z/OS SNMP client (command) setup and usage" on page 170 for information about how to use the **snmp** command. The SNMPv3 framework allows you to change the SNMP agent's configuration dynamically. Some example operations supported are:

- Change the security keys for SNMP managers.
- Add new users or SNMP managers.
- Change the group definition.

Performance statistics displays and monitoring

Refer to 4.4.2, "z/OS SNMP client (command) setup and usage" on page 170 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. Some possible monitoring options are:

- Monitor the throughput.
- Monitor the transmission delay.
- Monitor the amount of data that meet a policy specification.

Problem determination tasks for the z/OS SNMP agent

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 it syntax, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Some MIB values are very 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:

- ifInErrors, ifOutErrors: give the number of inbound/outbound frames that were discarded due to errors.
- ifInDiscards, ifOutDiscards: give the number of inbound/outbound frames that were discarded due to resource limitations.

Examples of MIB objects that can be used to determine problems in the IP layer are:

- 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 due to resource limitations.
- ipReasmFails: provides the number of failures detected by the IP re-assembly algorithm (for whatever reason: timed out, errors, etc).

The following 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.

4.4.2 z/OS SNMP client (command) setup and usage

The two SNMP client applications provided with z/OS Communications Server are:

- SNMP command from the NetView environment
- **snmp** command in the z/OS shell

The SNMP command in the NetView environment requires the use of the NetView product. It supports SNMP version 1. The **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, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Setup preparation and command usage are discussed in the following sections:

- Implementation for the SNMP NetView command
- Implementation of the osnmp z/OS UNIX command
- Using the osnmp/snmp z/OS UNIX command

Implementation for the SNMP NetView 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. We discuss three of those here:

- SNMP Query Engine started task
- SNMPARMS control member for NetView
- MIBDESC.DATA data set

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. Refer to Example 4-8 on page 163.

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. Refer to Example 4-9 on page 163.

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).

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 4-21.

Example 4-21 /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     9.67.113.79 snmpv2c
sc30b    10.20.1.230 snmpv2c
sc31b    10.20.1.241 snmpv2c
sc32b    10.20.1.221 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 and modify it for your installation.

Using the osnmp/snmp z/OS UNIX command

Refer to *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:

- Example 4-22 shows system information.
- Example 4-23 on page 173 shows stack information.
- Example 4-24 on page 173 shows omproute/ospf information.
- Example 4-25 on page 174 shows TN3270 information.

Tip: If you are only interested in a single MIB object, then you should use the **get** option with the fully qualified MIB object, instead of the **getbulk** option as shown in our examples.

Example 4-22 snmp system information

```
CS06 @ SC30:/u/cs06>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 V1R8 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 = 1.3.6.1.4.1.2.11.7.3

CS06 @ SC30:/u/cs06>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 V1R8 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
CS06 @ SC30:/u/cs06>
```

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 10.20.1.230 and the Foreign socket would be 10.20.1.241, as seen in Example 4-23.

Example 4-23 snmp stack information

```
CS06 @ SC30:/u/cs06>snmp -h sc30b -c j0s9m2ap -v getbulk ibmMvsTcpipProcname
ibmMvsTcpipProcname.0 = TCPIPB
ibmMvsTcpipAsid.0 = 111

CS06 @ SC30:/u/cs06>snmp -h sc31b -c j0s9m2ap -v getbulk ibmMvsTcpipProcname
ibmMvsTcpipProcname.0 = TCPIPB
ibmMvsTcpipAsid.0 = 132

CS06 @ SC30:/u/cs06>snmp -h sc30b -c j0s9m2ap -v -m 30 getbulk tcpConnectionEntry
tcpConnectionState.1.4.10.20.1.230.23.1.4.10.20.1.241.1029 = 5
tcpConnectionState.1.4.10.20.1.230.1056.1.4.10.20.1.230.1055 = 8
tcpConnectionState.1.4.10.20.1.230.1062.1.4.10.20.1.230.1061 = 8
tcpConnectionState.1.4.10.20.10.21.992.1.4.10.20.5.226.1035 = 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

CS06 @ SC30:/u/cs06>snmp -h sc31b -c j0s9m2ap -v -m 30 getbulk tcpConnectionEntry
tcpConnectionState.1.4.10.20.1.241.1029.1.4.10.20.1.230.23 = 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 4-24 snmp Omproute/OSPF information

```
CS06 @ SC30:/u/cs06>snmp -h sc30b -c j0s9m2ap -v -m 10 getbulk ospf
ospfRouterId.0 = 10.20.1.230
ospfAdminStat.0 = 1
ospfVersionNumber.0 = 2

CS06 @ SC30:/u/cs06>snmp -h sc31b -c j0s9m2ap -v -m 10 getbulk ospf
ospfRouterId.0 = 10.20.1.241
ospfAdminStat.0 = 1
ospfVersionNumber.0 = 2

CS06 @ SC30:/u/cs06>snmp -h sc30b -c j0s9m2ap -v -m 10 getbulk ospfIfTable
ospfIfIpAddress.10.20.1.230.0 = 10.20.1.230
ospfIfIpAddress.10.20.2.232.0 = 10.20.2.232
ospfIfIpAddress.10.20.2.234.0 = 10.20.2.234
ospfIfIpAddress.10.20.3.233.0 = 10.20.3.233
ospfIfIpAddress.10.20.3.235.0 = 10.20.3.235
ospfIfIpAddress.10.20.4.234.0 = 10.20.4.234
ospfIfIpAddress.10.20.4.235.0 = 10.20.4.235
ospfIfIpAddress.10.20.5.236.0 = 10.20.5.236
ospfIfIpAddress.10.20.10.21.0 = 10.20.10.21
ospfIfIpAddress.10.20.10.22.0 = 10.20.10.22
```

```
CS06 @ SC30:/u/cs06>snmp -h sc31b -c j0s9m2ap -v -m 10 getbulk ospfIfTable
ospfIfIpAddress.10.20.1.241.0 = 10.20.1.241
ospfIfIpAddress.10.20.2.242.0 = 10.20.2.242
ospfIfIpAddress.10.20.2.244.0 = 10.20.2.244
ospfIfIpAddress.10.20.3.243.0 = 10.20.3.243
ospfIfIpAddress.10.20.3.245.0 = 10.20.3.245
ospfIfIpAddress.10.20.4.244.0 = 10.20.4.244
ospfIfIpAddress.10.20.4.245.0 = 10.20.4.245
ospfIfIpAddress.10.20.5.246.0 = 10.20.5.246
ospfIfIpAddress.10.20.10.31.0 = 10.20.10.31
ospfIfIpAddress.10.20.10.32.0 = 10.20.10.32
```

Example 4-25 snmp TN3270E Telnet server information

```
CS06 @ SC30:/u/cs06>snmp -h sc30b -c j0s9m2ap -v -m 40 getbulk ibmMvsTN3270ConnTable
ibmMvsTN3270ConnStartTime.1.4.10.20.1.230.23.1.4.10.20.1.241.1031=2006-10-25,0:0:58.8
ibmMvsTN3270ConnAppl.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = SC30TS02
ibmMvsTN3270ConnLuName.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = SC30BB01
ibmMvsTN3270ConnLogMode.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = SNX32704
ibmMvsTN3270ConnProto.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = '02'h
ibmMvsTN3270ConnRtGroupIndex.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 1
ibmMvsTN3270ConnRtIpMethod.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtAvgRt.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtAvgIpRt.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtAvgCountTrans.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 8
ibmMvsTN3270ConnRtIntTimeStamp.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 2006-10
-25,0:6:58.8
ibmMvsTN3270ConnRtTotalRts.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtTotalIpRts.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtCountTrans.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 12
ibmMvsTN3270ConnRtCountIP.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtElapsRndTrpSq.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtElapsIpRtSq.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtElapsSnaRtSq.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtBucket1Rts.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 12
ibmMvsTN3270ConnRtBucket2Rts.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtBucket3Rts.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtBucket4Rts.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 0
ibmMvsTN3270ConnRtBucket5Rts.1.4.10.20.1.230.23.1.4.10.20.1.241.1031 = 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
CS06 @ SC30:/u/cs06>
```

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 4-26.

Example 4-26 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 4-27.

Example 4-27 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 10.20.10.21   ENDDESTIPGROUP
DESTIPGROUP ADMIN       10.20.10.22   ENDDESTIPGROUP
DESTIPGROUP PAYROLL     10.20.10.23   ENDDESTIPGROUP
DESTIPGROUP SHIPPING    10.20.1.230   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 Communication 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 Communication Server V1R8 added two new counters to 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. Another object added to the IBM MVSTCP/IP enterprise-specific MIB module is the following:

- **ibmMvsCpcNd -**The Central Processing Complex node descriptor value for the central processing complex on which the TCP/IP subagent is active. This value can be used to uniquely identify this CPC. If the node descriptor cannot be obtained, then this MIB object will be set to a zero-length string.

Example 4-28 shows some examples that we obtained from **snmp** commands in Unix System Services.

Example 4-28 Examples of TCP layer accepted connections counters.

```
CS06 @ SC30:/u/cs06>snmp -h 10.40.1.230 -c public -v getbulk ibmMvsTcpHCAcceptCo
unt
ibmMvsUdpLastAct.0.0.0.0.0.161 = 0
CS06 @ SC30:/u/cs06>snmp -h 10.40.1.230 -c public -v getbulk ibmMvsTcpAcceptCoun
t
ibmMvsTcpAcceptCount.0 = 1
CS06 @ SC30:/u/cs06>cd \
```

In the Example 4-29 we see that the object ibmMvsCpcNd can be obtained in two different ways. One is through the snmp command and the other through the MVS console command D M=CPU.

Example 4-29 The ibmMvsCpcNd obtained in two different ways.

```
CS06 @ SC30:/u/cs06>snmp -h 10.40.1.230 -c public -v getbulk ibmMvsCpcNd
ibmMvsCpcNd.0 = 'f0f0f2f0f9f4e2f1f8c9c2d4f0f2f0f0f0f0f0f0f0f2f9f9f1c5fff0'h
CS06 @ SC30:/u/cs06>
```

```
D M=CPU
 IEE174I 15.01.12 DISPLAY M 137
 PROCESSOR STATUS
 ID  CPU                SERIAL
 00  +                   23991E2094
 01  +                   23991E2094
 02  +                   23991E2094
 03  +                   23991E2094

 CPC ND = 002094.S18.IBM.02.00000002991E
 CPC SI = 2094.712.IBM.02.000000000002991E
 CPC ID = 00
 CPC NAME = SCZP101
 LP NAME = A23        LP ID = 23
 CSS ID  = 2
 MIF ID  = 3

 + ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
 N NOT AVAILABLE

 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
```

Removal of DPI API 4KB buffer restriction

In previous versions we had 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 byte SNMP for responses.

Important: For subagents which are not provided by z/OS Communication Server, the subagent load modules must be linked again to include the updated DPI API functions.

As a result, the applications can retrieve more data per SNMP request, especially when using the SNMP GETBULK and BULKWALK options.

Changes for message EZZ6317I

In z/OS V1R4, 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 Communication Server agent to network management applications.

This implementation now enables the message EZZ6317I to be written to the console and syslog daemon, and is more visible to those customers currently configuring this MIB object.

Sample of OSNMPD.DATA information is shipped with the z/OS Communication 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 4-30 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 and syslog daemon as well.

Example 4-30 OSNMPD.DATA without sysObjectId.

```
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5694-A01
# (C) Copyright IBM Corp. 1996, 2006
# Status = CSV1R8
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "Unknown"
  sysLocation "Unknown"
  sysName "z/OS V1R8 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 Communication 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 4-31 is before changing the OSNMPD.DATA file. We did not receive the EZZ6317I message.

Example 4-31 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
```

After changing the OSNMPD.DATA file, which enabled the sysObjectId (see Example 4-32).

Example 4-32 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
```

IP printing

IP printing supports the sending of print output from a client device on an IP network to a printer on an IP network. This chapter focuses on the IP printing functions that are available in the z/OS V1R8.0 Communications Server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains comprehensive descriptions of the individual parameters for setting up an LPD server. It also includes step-by-step checklists and supporting examples. It is not the intent of this book to duplicate the information in the manual, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 5.1.2, “Additional information sources for IP printing” on page 183.

This chapter discusses the following.

Section	Topic
5.1, “Overview” on page 182	Discusses the basic concepts of IP printing
5.2, “Why IP printing is important” on page 184	Discusses key characteristics of IP printing and why it may be important in your environment
5.3, “The common design scenarios for IP printing” on page 184	Presents commonly implemented IP printing design scenarios, their dependencies, advantages, considerations, and our recommendations
5.4, “How the printing functions are implemented” on page 189	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

5.1 Overview

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.

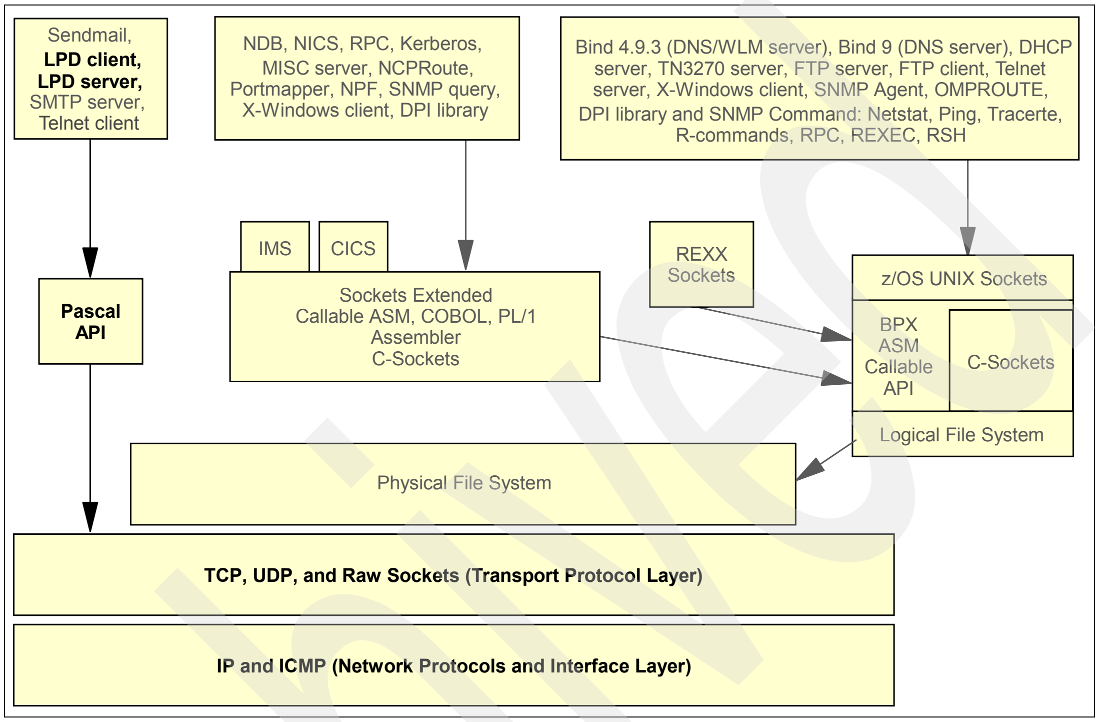

Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270 server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
REXX Sockets
z/OS UNIX Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

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.

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. The basic capabilities of LPR are very 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 your 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.

5.1.1 Basic concepts of IP printing

The diagram in Figure 5-2 on page 183 shows the relationship of LPR to LPD.

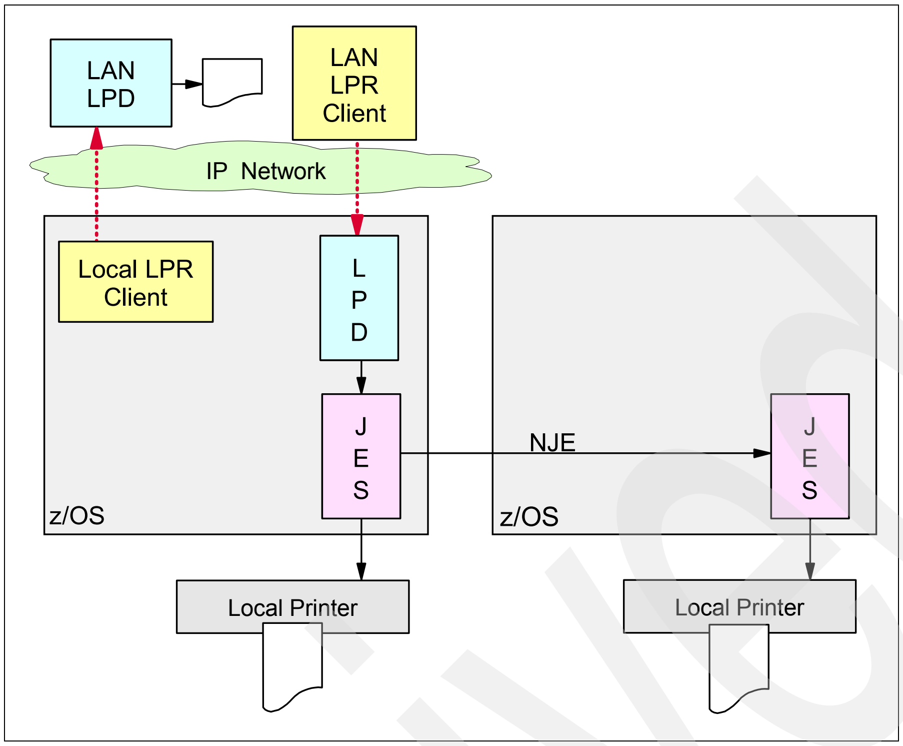


Figure 5-2 LPR/LPD relationships

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 once 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.

5.1.2 Additional information sources for IP printing

Detailed information for IP printing can be found in the following documents:

- *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.

5.2 Why IP printing is important

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 are discussed in 5.3.2, "Infoprint Server" on page 185 and 5.4.2, "Infoprint Server" on page 199.

5.3 The common design scenarios for IP printing

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. These scenarios are discussed in:

- LPR/LPD

- Infoprint Server

5.3.1 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.

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.

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.

Advantages of LPR/LPD

This is a simple implementation and does not require much planning.

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.

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 very seldom has anything to show the client. Once the file is under JES control, the LPD considers the file as having been delivered.

5.3.2 Infoprint Server

This section provides an overview of IP printing with the Infoprint Server. For additional information about the Infoprint Server, see:

- The 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

Description of 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.

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.

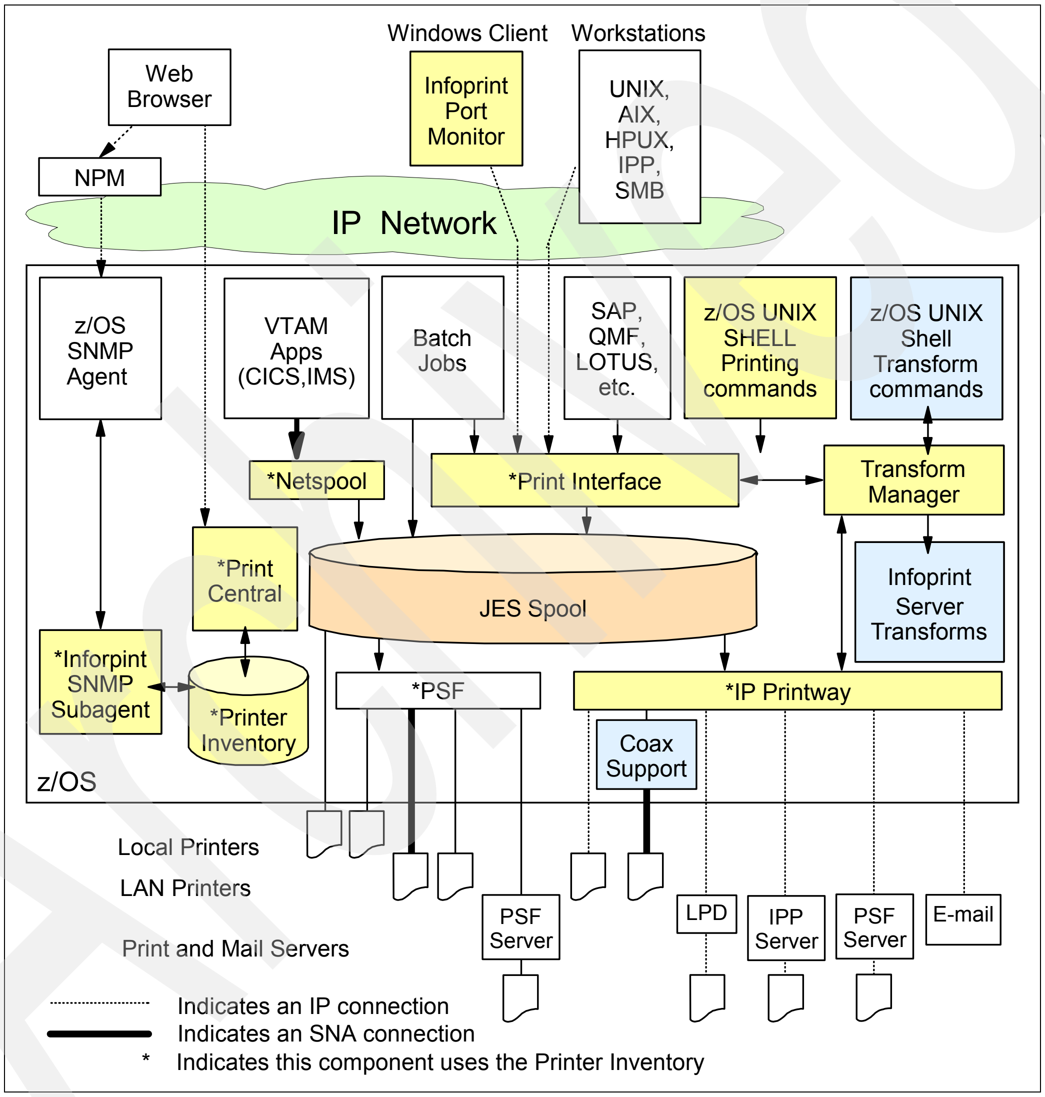


Figure 5-3 Infoprint components

The Infoprint Server Transforms components include:

- z/OS UNIX Shell Transform commands
- Infoprint Server Transforms
- Coax support

Following is a description of each component.

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 (HFS or zFS) that contain information about each printer and e-mail destination. The Printer Inventory also contains system configuration information for IP PrintWay™ and 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 CICS and 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 e-mail 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.

Dependencies of 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.

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 while improving manageability, data retrievability, and usability. For a complete discussion of these benefits, see 5.4.2, “Infoprint Server” on page 199.

Other advantages of the Infoprint Server are:

- 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 e-mailing.
- IBM Infoprint XT Extender for z/OS, a separate product, lets you transform Xerox files to AFP format for printing or e-mailing. 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 is comprised 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 in order to achieve a successful implementation.

5.3.3 Recommendations for IP printing

Based on the above discussions, we recommend several approaches that could meet your IP printing requirements.

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 may be faced with a requirement to reformat print files before they are delivered to printers requiring special data streams. A printing solution like the IBM Infoprint Server should be considered.

5.4 How the printing functions are implemented

Based upon the discussion in 5.3.3, “Recommendations for IP printing” on page 188, we show implementation details for the following scenarios.

- LPR/LPD
- 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 5.3.2, “Infoprint Server” on page 185.

As part of any implementation effort two appendixes in this book should be beneficial in planning your work:

- Environment variables are categorized by application in Appendix A, “Environment variables” on page 319.
- Sample files for each application are listed in Appendix B, “Sample files provided with TCP/IP” on page 331.

5.4.1 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 may be a local JES system printer, or it may 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.

The following topics discuss the implementation of LPR/LPD:

- Implementation tasks for LPR/LPD
- Configuration examples for LPR/LPD
- Verification for LPR/LPD
- Problem determination for LPR/LPD

Implementation tasks for LPR/LPD

The *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 steps to configure the LPD server are:

1. Specify AUTOLOG and PORT statements in the TCP/IP profile data set shipped as SYS1.SEZAINST(TCPDATA).
2. Update the LPD server cataloged procedure shipped as SYS1.SEZAINST(LPSPROC), but with an actual procedure name of LPSERVE.
3. Update the LPD server configuration data set shipped as SYS1.SEZAINST(LPDDATA).
4. Create a banner page (optional).

Configuration examples for LPR/LPD

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
```

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.)
```

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(LPDATB30) - 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 lpt1@ITSOPRT.TCP.RALEIGH.IBM.COM
; FAILEDJOB MAIL
;
SERVICE pun1 PUNCH
   LOCAL
   FILTERS l
   LINESIZE 80
;
;
OBEY CS06 CS07
```

Verification for LPR/LPD

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 5-4.

Example 5-4 LPQ samples

```
LPQ (ALL PRINTER testprt1 AT lpdsrvr1
LPQ (PRINTER testprt1 AT lpdsrvr1 TRACE
LPQ myuserid (PRINTER testprt1 AT lpdsrvr1
LPQ 273 (PRINTER testprt1 AT lpdsrvr1
```

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. An example is shown in Example 5-5.

Example 5-5 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 5-6.

Example 5-6 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 5-7.

Example 5-7 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@192.167.22.4
LPRSET (QUERY
EZB1020I Your LPR printer is currently set to testprt1 at 192.167.22.4.
```

If LPRSET has been used to set the default printer and server, then the other commands do not have to specify the printer or server, as in Example 5-8.

Example 5-8 Abbreviated LPR commands use the default setting

```
LPQ
LPR 'TESTFILE.PRINT'
LPRM 273
LPRM myjob
```

The startup messages for LPSERVE are shown in Example 5-9.

Example 5-9 LPSERVEB start up 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(LPDATB30)' PREFIX
TCPIP .
```

A display of the TCP/IP stack connections shows the LPSERVE listener in Example 5-10.

Example 5-10 LPSERVEB listening on port 515

```
SDSF ULOG  CONSOLE CS06                          LINE 164     COLUMNS 44- 123
 COMMAND INPUT ===>                                           SCROLL ===> CSR
-d tcpip,tcpipb,n,conn
 EZD0101I NETSTAT CS V1R8 TCPIPB 667
 USER ID  CONN     STATE
 DCD1DIST 00004244 LISTEN
   LOCAL SOCKET:   0.0.0.0..38241
   FOREIGN SOCKET: 0.0.0.0..0
 DCD1DIST 00004241 LISTEN
   LOCAL SOCKET:   0.0.0.0..38240
   FOREIGN SOCKET: 0.0.0.0..0
 LDAPDBCG 00003096 LISTEN
   LOCAL SOCKET:   0.0.0.0..33389
   FOREIGN SOCKET: 0.0.0.0..0
 LPSERVEB 00004394 LISTEN
   LOCAL SOCKET:   0.0.0.0..515
   FOREIGN SOCKET: 0.0.0.0..0
 OMPB     00000026 ESTBLSH
   LOCAL SOCKET:   127.0.0.1..1026
   FOREIGN SOCKET: 127.0.0.1..1027
 RXSERVEB 0000273C CLOSWT
   LOCAL SOCKET:   10.20.1.230..1062
   FOREIGN SOCKET: 10.20.1.230..1061
 RXSERVEB 0000265F CLOSWT
   LOCAL SOCKET:   10.20.1.230..1056
   FOREIGN SOCKET: 10.20.1.230..1055
 RXSERVEB 00002640 LISTEN
   LOCAL SOCKET:   0.0.0.0..512
   FOREIGN SOCKET: 0.0.0.0..0
 RXSERVEB 00002641 LISTEN
   LOCAL SOCKET:   0.0.0.0..514
   FOREIGN SOCKET: 0.0.0.0..0
 TCPIPB   00000025 ESTBLSH
   LOCAL SOCKET:   127.0.0.1..1027
   FOREIGN SOCKET: 127.0.0.1..1026
 TN3270B  00000033 LISTEN
   LOCAL SOCKET:   ::..992
   FOREIGN SOCKET: ::..0
 TN3270B  00000034 LISTEN
   LOCAL SOCKET:   ::..23
   FOREIGN SOCKET: ::..0
```

Problem determination for LPR/LPD

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 may not allow the LPR client program access to the stack.
  – LPD may not be configured, or may 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.

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 5-11.

Example 5-11 LPSERVE SMSG command responses

```
SDSF OUTPUT DISPLAY LPSERVEB STC07362  DSID   103 LINE 0       COLUMNS 02- 81
***** 10/19/05 *****
12:47:30 EZB0831I IBM MVS LPD Version CS V1R8 on 10/19/05 at 12:47:30
12:47:30 EZB0832I
12:47:30 EZB0857I Using hardcoded translation tables
12:47:50 EZB0705I 10/19/05 12:47:50
12:47:50 EZB0834I Ready
14:55:03 EZB0786I Command received "TRACE ON".
14:55:03 EZB0789I GetNextNote with ShouldWait of TRUE
14:55:09 EZB0790I GetNextNote returns.  Connection 0 Notification SMSG received
14:55:09 EZB0786I Command received "TRACE OFF".
14:55:21 EZB0786I Command received "PRINT WORK".
14:55:21 EZB0731I       Work Queue start
14:55:21 EZB0733I       Work Queue end
```

On the TSO ISPF shell command line we issued the LPR command:

```
   lpr ‘tcpipb.tcpparms(datab30)’ at 10.20.1.230 printer sysprt1 class h
```

We received the error message shown in Example 5-12 from the LPR command.

Example 5-12 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 = 10.20.1.230
 ***
```

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 5-13.

Example 5-13 Port reservation list indicates LPR ports missing (722-731)

```
d tcpip,tcpipb,n,portlist
EZD0101I NETSTAT CS V1R8 TCPIPB 789
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: 10.20.10.241
00023 TCP  TN3270B  DU
00025 TCP  SMTPB    DA
00053 TCP  NAMED9B  DA
00111 TCP  PORTMAPB DA
00512 TCP  OMVS     DABU
      BINDSPECIFIC: 10.20.10.242
00512 TCP  RXSERVEB DAU
00514 TCP  RXSERVEB DAU
00514 TCP  OMVS     DABU
      BINDSPECIFIC: 10.20.10.242
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 5-14.

Example 5-14 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 5-15 on page 196.

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 5-15 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
 EZD0101I NETSTAT CS V1R8 TCPIPB 023
 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: 10.20.10.241
 00023 TCP  TN3270B  DU
 00025 TCP  SMTPB    DA
 00053 TCP  NAMED9B  DA
 00111 TCP  PORTMAPB DA
 00512 TCP  OMVS     DABU
       BINDSPECIFIC: 10.20.10.242
 00512 TCP  RXSERVEB DAU
 00514 TCP  RXSERVEB DAU
 00514 TCP  OMVS     DABU
       BINDSPECIFIC: 10.20.10.242
 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
 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
 38 OF 38 RECORDS DISPLAYED
 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 *before* our LPR command, as seen in Example 5-16.

Example 5-16 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-
      CS064    JOB07178 CS06      144 H HOLD  LOCAL                   46
      CS065    JOB07180 CS06      144 H HOLD  LOCAL                   46
      CS06     TSU07131 CS06      144 S HOLD  LOCAL                  276
      CS06     TSU07132 CS06      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(datab30)' at 10.20.1.230 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, and we do not show all of it here. However, we do include some important lines from the trace that show the progression of servicing the received print file from LPR. The trace is shown in Example 5-17.

Example 5-17 LPD trace receiving and processing a file from LPR

```
15:27:24 EZB0786I Command received "TRACE ON".
15:27:24 EZB0789I GetNextNote with ShouldWait of TRUE
16:22:05 EZB0790I GetNextNote returns.  Connection 0 Notification Connection sta
16:22:05 EZB0779I New connection state Open (8673) on connection 0 with reason 0
16:22:05 EZB0626I Allocated ConnectionBlock at 001C7E08
16:22:05 EZB0627I Passive open on port 515
.............................
16:22:05 EZB0716I Job 115 received sysprt1 10.20.1.230
16:22:05 EZB0734I Job 115 added to work queue
16:22:05 EZB0716I Job 115 scheduled sysprt1 10.20.1.230
16:22:05 EZB0776I Released StepBlock at 000060C8
16:22:05 EZB0777I Released ConnectionBlock at 001BFE08
16:22:05 EZB0824I ProcessWork starting on job queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBstartPRINTING
```

```
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0825I    Job 115 for sysprt1 dispatched in state JOBstartPRINTING
16:22:05 EZB0716I Job 115 printing sysprt1 10.20.1.230
16:22:05 EZB0827I ProcessWork end with queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBcontinuePRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0789I GetNextNote with ShouldWait of FALSE
16:22:05 EZB0824I ProcessWork starting on job queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBcontinuePRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0825I    Job 115 for sysprt1 dispatched in state JOBcontinuePRINTING
16:22:05 EZB0827I ProcessWork end with queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBfinishPRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0789I GetNextNote with ShouldWait of FALSE
16:22:05 EZB0824I ProcessWork starting on job queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBfinishPRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0825I    Job 115 for sysprt1 dispatched in state JOBfinishPRINTING
16:22:05 EZB0716I Job 115 sent sysprt1 10.20.1.230
16:22:05 EZB0769I Job 115 removed from work queue
16:22:05 EZB0751I Released StepBlock at 00006280
16:22:06 EZB0716I Job 115 purged sysprt1 10.20.1.230
16:22:06 EZB0771I Released JobBlock at 000AE960
16:22:06 EZB0827I ProcessWork end with queue
16:22:06 EZB0731I       Work Queue start
16:22:06 EZB0733I       Work Queue end
16:22:06 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 10.20.1.230 printer sysprt1 class h
```

When the trace is off, the output in LPSERVEB's SYSPRINT appears as it does in Example 5-18.

Example 5-18 LPSERVEB normal processing messages when trace is off

```
16:35:12 EZB0716I Job 88 received sysprt1 10.20.1.230
16:35:12 EZB0716I Job 88 scheduled sysprt1 10.20.1.230
16:35:12 EZB0716I Job 88 printing sysprt1 10.20.1.230
16:35:12 EZB0716I Job 88 sent sysprt1 10.20.1.230
16:35:12 EZB0716I Job 88 purged sysprt1 10.20.1.230
```

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 5-19.

Example 5-19 LPSERVEB entries in the JES output queue

```
SDSF OUTPUT ALL CLASSES ALL FORMS     LINES 441,440    LINE 47-58 (58)
```

```
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
s    LPSERVEB STC07362 TCPIP     144 H STD      LOCAL                   13
s    LPSERVEB STC07362 TCPIP     144 H STD      LOCAL                   13
     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
```

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 5-20.

Example 5-20 LPSERVEB contents of the two print files waiting to be printed

```
SDSF OUTPUT DISPLAY LPSERVEB STC07362  DSID   111 LINE 0       COLUMNS 02- 81
WTSC30B.ITSO.IBM.COM:TCPIPB.TCPPARMS(DATAB30)

TCPIPJOBNAME TCPIPB
HOSTNAME WTSC30B
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  9.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

SDSF OUTPUT DISPLAY LPSERVEB STC07362  DSID   112 LINE  DATA SET DISPLAYED
WTSC30B.ITSO.IBM.COM:TCPIPB.TCPPARMS(DATAB31)

TCPIPJOBNAME TCPIPB
HOSTNAME WTSC31B
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  9.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
```

5.4.2 Infoprint Server

Because we cannot cover the detailed implementation of the Infoprint Server in this book, please refer to the reference material listed in 5.3.2, “Infoprint Server” on page 185 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 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 ▸ E-mail	▸ 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
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 on page 201 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 has a configuration file, as shown in Example 5-21.

Example 5-21 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
```

Message control can be customized as shown in Example 5-22.

Example 5-22 Certain messages can be selected for syslog

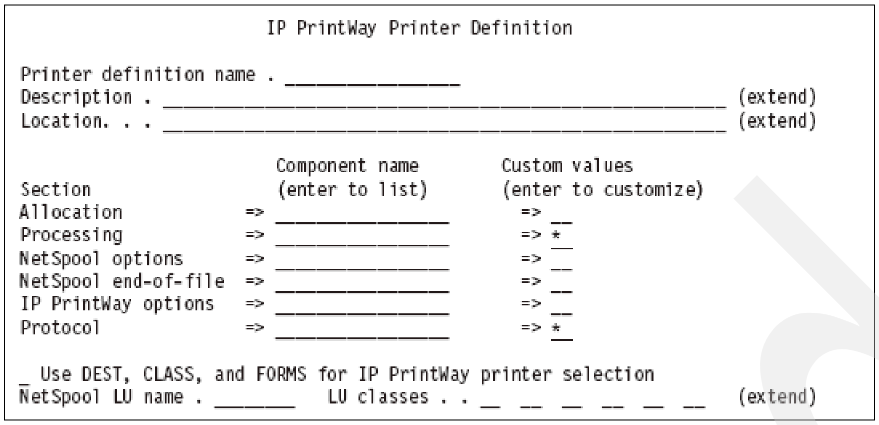
```
# aopmsg.conf - Infoprint Server Message Configuration file
#-----------------------------------------------------------------------------
hardcopy-messages = list
hardcopy-message-list = {AOP3614I AOP3803E}
#hardcopy-messages = all
#hardcopy-messages = none
```

Sample 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.

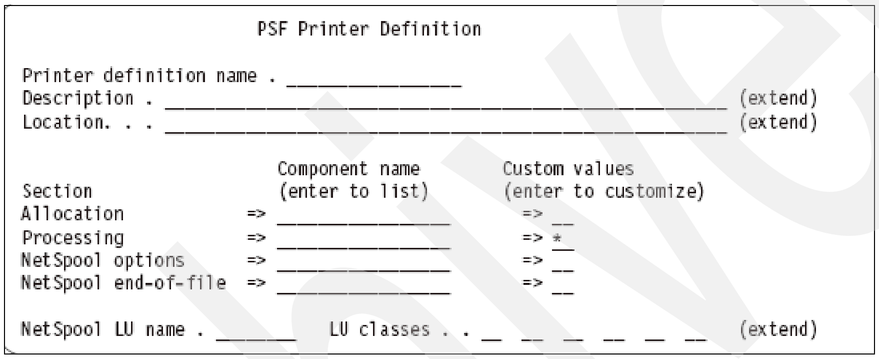
Some sample panels are shown here:

- Main ISPF panel for an IP PrintWay printer definition (Figure 5-4 on page 203)
- Main ISPF panel for a PSF printer definition (Figure 5-5 on page 203)
- Main ISPF panel for a General printer definition (Figure 5-6 on page 203)
- ISPF panel for the NetSpool Options component (Figure 5-7 on page 204)
- ISPF panel for the IP PrintWay Options section or component (Figure 5-8 on page 204)
- LPR Protocol panel (Figure 5-9 on page 205)
- VTAM Protocol panel (Figure 5-10 on page 205)
- E-mail Protocol panel (Figure 5-11 on page 205)
- ISPF panel for an IP PrintWay FSS definition (Figure 5-12 on page 206)



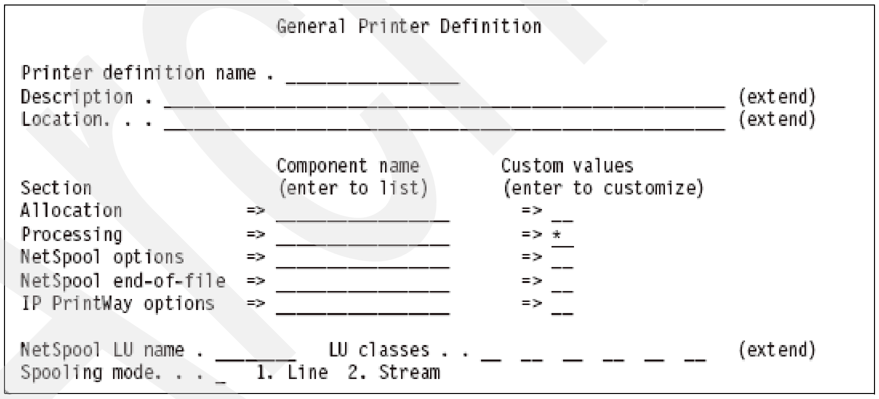
```
                          IP PrintWay Printer Definition

 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



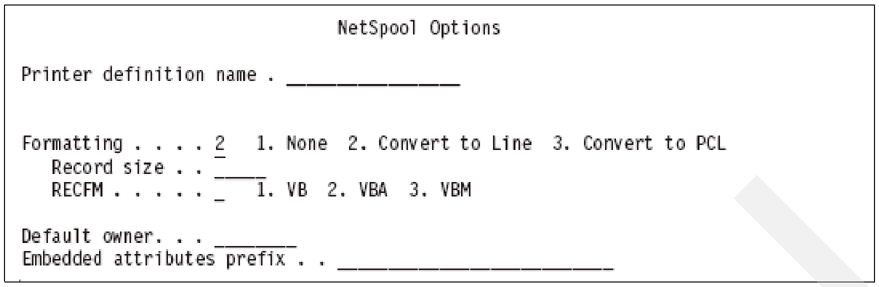
```
                         PSF Printer Definition

 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



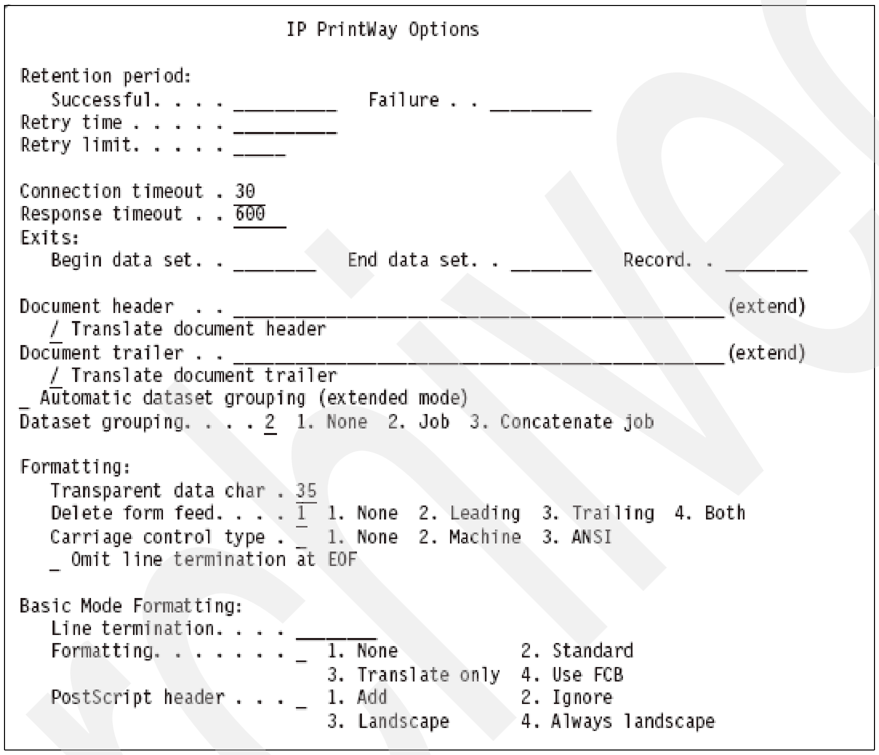
```
                           General Printer Definition

 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



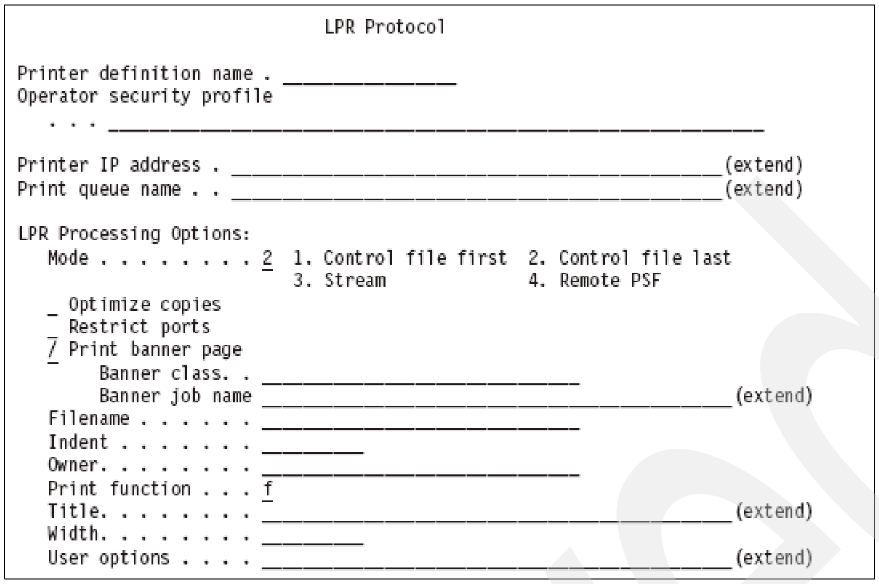
```
                              NetSpool Options

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



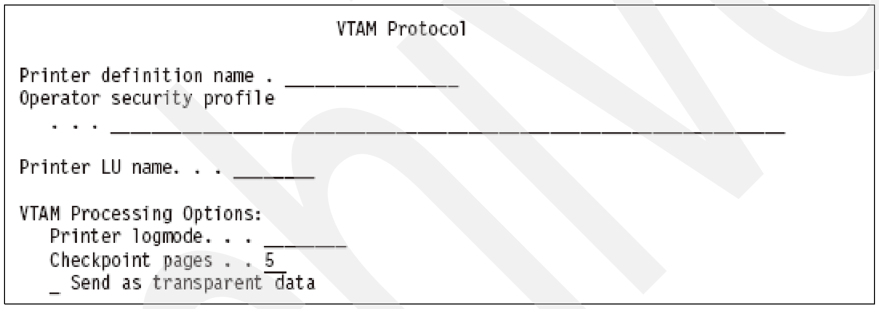
```
                          IP PrintWay Options

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



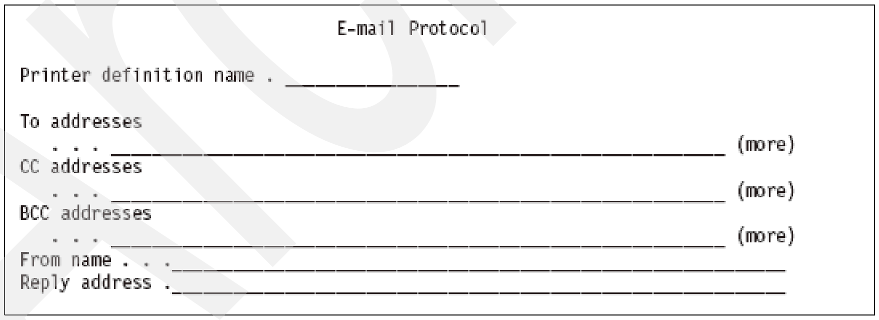
```
                              LPR Protocol

Printer definition name . ________________
Operator security profile
   . . . ______________________________________________________________

Printer IP address . ______________________________________________(extend)
Print queue name . . ______________________________________________(extend)

LPR Processing Options:
   Mode . . . . . . . . 2  1. Control file first  2. Control file last
                           3. Stream              4. Remote PSF
   _ Optimize copies
   _ Restrict ports
   / Print banner page
        Banner class. . ______________________________
        Banner job name ____________________________________________(extend)
   Filename . . . . . . ______________________________
   Indent . . . . . . . _________
   Owner. . . . . . . . ______________________________
   Print function . . . f
   Title. . . . . . . . ____________________________________________(extend)
   Width. . . . . . . . _________
   User options . . . . ____________________________________________(extend)
```

Figure 5-9   ISPF LPR Protocol panel



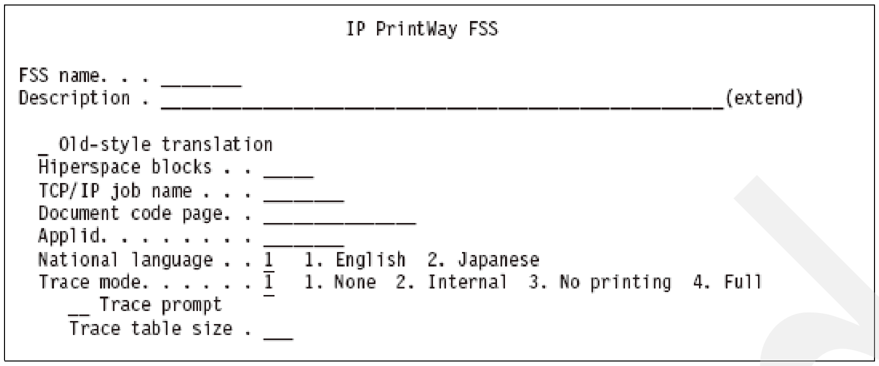
```
                              VTAM Protocol

Printer definition name . ________________
Operator security profile
   . . . ________________________________________________________________

Printer LU name. . . ________

VTAM Processing Options:
   Printer logmode. . . ________
   Checkpoint pages . . 5
   _ Send as transparent data
```

Figure 5-10   ISPF VTAM Protocol panel



```
                              E-mail Protocol

Printer definition name . ________________

To addresses
   . . . __________________________________________________________ (more)
CC addresses
   . . . __________________________________________________________ (more)
BCC addresses
   . . . __________________________________________________________ (more)
From name . . .__________________________________________________________
Reply address .__________________________________________________________
```

Figure 5-11   ISPF E-mail Protocol panel



```
                                IP PrintWay FSS

FSS name. . . _______
Description . _______________________________________________(extend)

  _ Old-style translation
  Hiperspace blocks . . ____
  TCP/IP job name . . . _______
  Document code page. . _____________
  Applid. . . . . . . . _______
  National language . . 1   1. English  2. Japanese
  Trace mode. . . . . . 1   1. None  2. Internal  3. No printing  4. Full
     __ Trace prompt
     Trace table size . ___
```

Figure 5-12   ISPF FSS printer definition panel

For complete implementation details on the Infoprint Server, refer to 5.3.2, “Infoprint Server” on page 185.

6 INETD

INETD, also known as the Internet super daemon, is an application that listens for connection requests on behalf of other applications. This chapter focuses on INETD functions that are available in the z/OS V1R8 Communications Server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

The *z/OS UNIX System Services Command Reference*, SA22-7802, and *z/OS UNIX System Services Planning*, GA22-7800, contain excellent descriptions of the individual parameters for setting up INETD. It is not the intent of this book to duplicate the information in the manual, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 6.1.2, "For additional information about INETD" on page 210.

This chapter discusses the following.

Section	Topic
6.1, "Overview" on page 208	Discusses the basic concepts of INETD
6.2, "Why INETD is important" on page 210	Discusses key characteristics of INETD and why it may be important in your environment
6.3, "The common design scenarios for INETD" on page 210	Presents commonly implemented INETD design scenarios, their dependencies, advantages, considerations, and our recommendations
6.4, "How INETD is implemented" on page 211	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

6.1 Overview

INETD is one of the standard applications provided with the z/OS Communications Server. It is a generic listener. It 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.

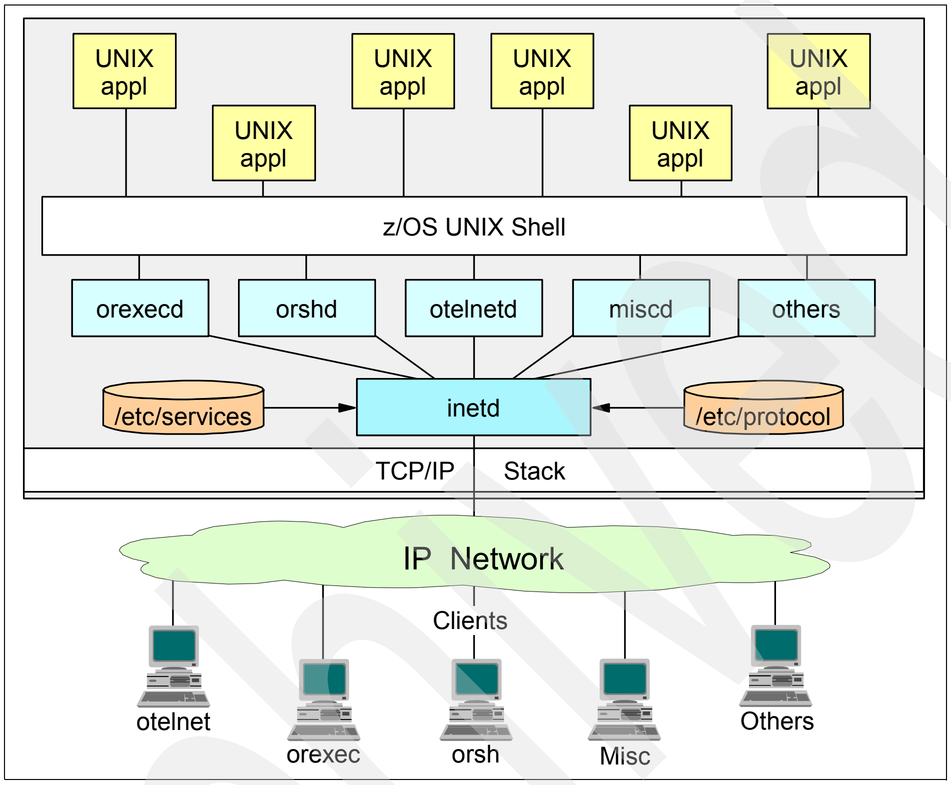


Figure 6-1 INETD and its supported applications

6.1.1 Basic concepts

The INETD servers provide access to the z/OS UNIX shell via 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; these are listed in Table 6-1.

Table 6-1 Applications that use INETD

Application	Description
z/OS UNIX Telnet Server	Refer to Chapter 8, “z/OS UNIX Telnet server” on page 247.
z/OS UNIX REXEC Server	Refer to Chapter 9, “Remote execution” on page 257.
z/OS UNIX RSH server	Refer to Chapter 9, “Remote execution” on page 257.
echo	Repeats any data received back to the sender.
discard	Throws away any received data.
chargen	Sends predefined or random data and throws 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.

Application	Description
popper	Refer to Chapter 7, "Mail servers" on page 217.

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 jobname INET, 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.

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 may 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 may be passed to the application.

INETD parameters

INETD can accept two command line parameters: -d and 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.

6.1.2 For additional information about INETD

For additional information, refer to:

- *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.

6.2 Why INETD is important

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.

6.3 The common design scenarios for INETD

INETD is a simple application with limited configuration options. We only show a single design scenario for INETD.

6.3.1 A simple INETD design

In this section we discuss a simple INET implementation using the more common configuration options. We provide a configuration file that starts all servers and internal services.

Dependencies

INETD only requires an active stack and an /etc/inetd.conf configuration file.

Advantages

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.

Considerations

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.

6.3.2 Recommendations

We recommend using INETD only if you want to start otelnetd, orexecd, orshd, or if you want to use the internal services of INETD for testing purposes. If INETD is needed, we recommend using the INETD design discussed in 6.3.1, “A simple INETD design” on page 210.

6.4 How INETD is implemented

Based upon our recommendations from 6.3.2, “Recommendations” on page 210, we show implementation details for the simple INETD design.

6.4.1 A simple INETD design

In our simple design we will configure INETD with all services active. If use of a particular service is not desired in your environment, simply comment out the statements for the undesired service.

Implementation tasks

The tasks are:

1. Create a configuration file.
2. Add port statements to ETC.SERVICES or /etc/services.
3. Ensure the PORT statement in PROFILE.TCPIP reserves the port for INETD.
4. Start INETD.

Creating a configuration file

First, copy the sample configuration file from /samples/inetd.conf to a new location such as /etc/inetd.conf.

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.

Finally, you will need to determine which user name INETD and the servers started by INETD will run under. We recommend using OMVSKERN. The user name should have read access to the BPX.DAEMON facility class in RACF.

Example 6-1 shows our 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
```

Port statements

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. Refer to our /etc/services file in Example 6-2.

Example 6-2 /etc/services

```
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

While not required, we recommend 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 10.20.10.251
    37   TCP   INETD1
    37   UDP   INETD1
    110  TCP   INETD1
    512  TCP   INETD1
    513  TCP   INETD1
    514  TCP   INETD1
```

Note: Refer to Chapter 8, “z/OS UNIX Telnet server” on page 247, for details on why port 23 might be bound to a specific IP address.

Start INETD

There are two methods for starting INETD: by shell script in /etc/rc or as a started task. We recommend starting the INETD daemon via a shell script in /etc/rc because causes INETD to be automatically started when z/OS UNIX is started. 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
//*
//INETD EXEC PGM=INETD,REGION=OK,TIME=NOLIMIT,
//      PARM='POSIX(ON) ALL31(ON)/'
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Verification

The best method to verify that INETD is running correctly is to issue a **`netstat`** command with no parameters. If INETD is started correctly, the output of the **`netstat`** command should be similar to the output shown in Example 6-6.

Example 6-6 Output of netstat after INETD is started

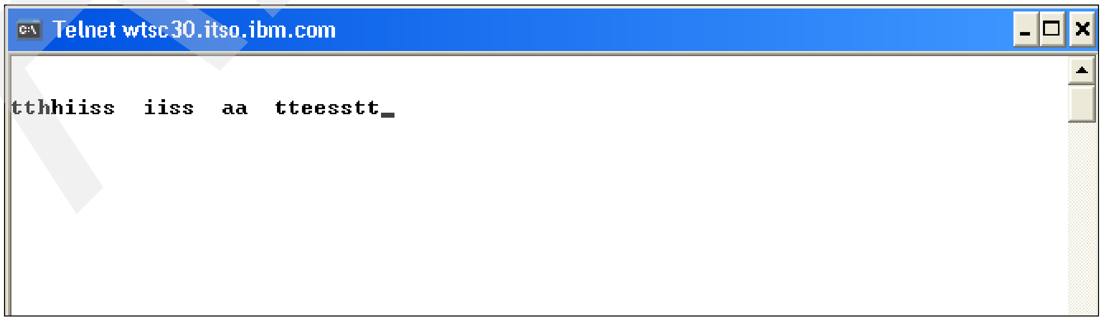
```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIP          17:50:53
User Id  Conn     State
-------  ----     -----

INETD1   00000049 Listen
  Local Socket:   9.12.4.212..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:   9.12.4.212..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:   9.12.4.212..110
  Foreign Socket: 0.0.0.0..0
INETD1   0000004B Listen
  Local Socket:   9.12.4.212..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 INET1, 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, netstat does not verify that each of the servers is working. We will 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 257, and otelnetd is discussed in Chapter 8, “z/OS UNIX Telnet server” on page 247.

Verifying 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.



```
tthhiiss  iiss  aa  tteesstt_
```

Figure 6-2 Verification of the echo service

Verifying 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.

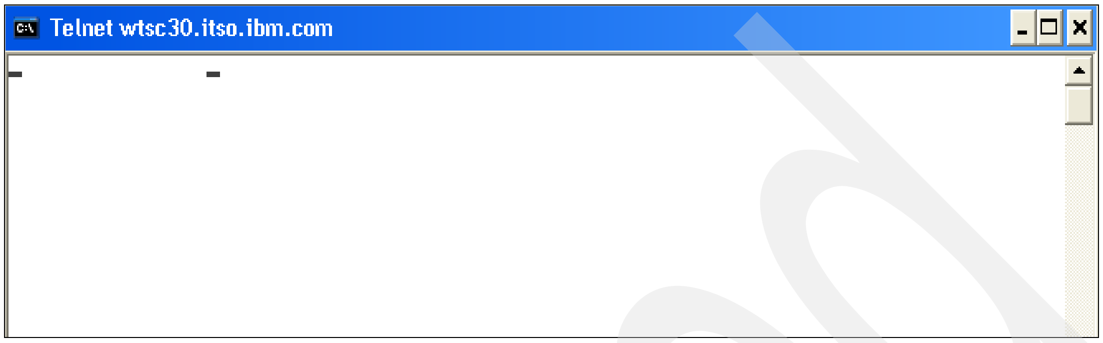
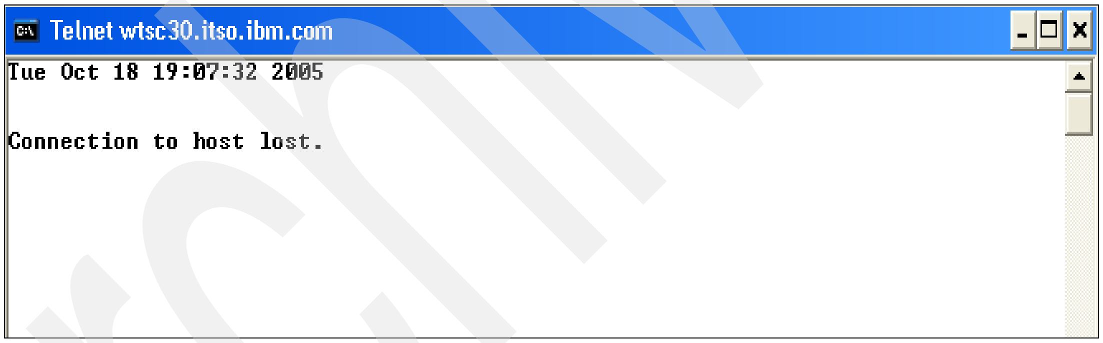


Figure 6-3 Verification of the discard service

Verifying 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.

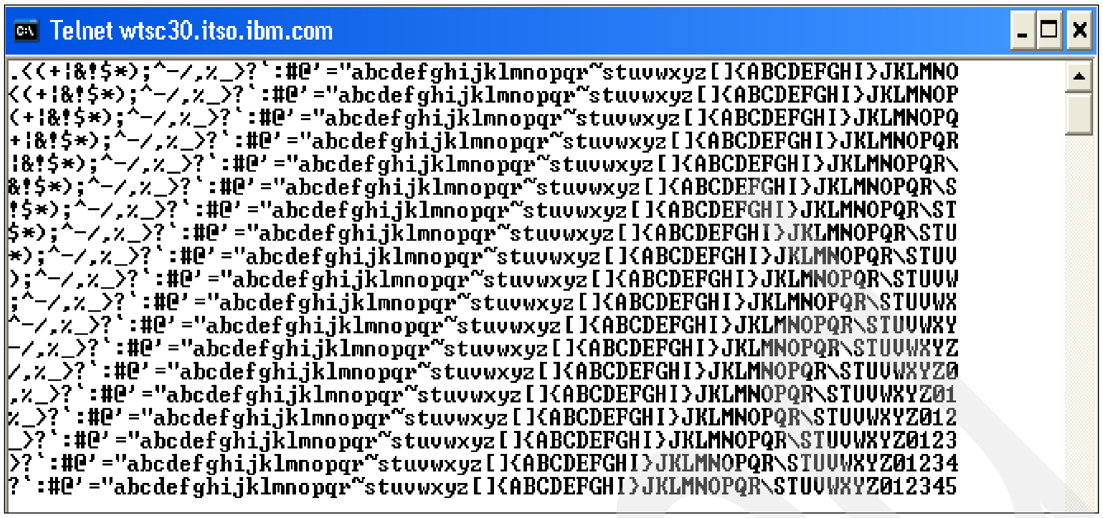


```
Tue Oct 18 19:07:32 2005


Connection to host lost.
```

Figure 6-4 Verification of the daytime service

Verifying 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 on page 216.

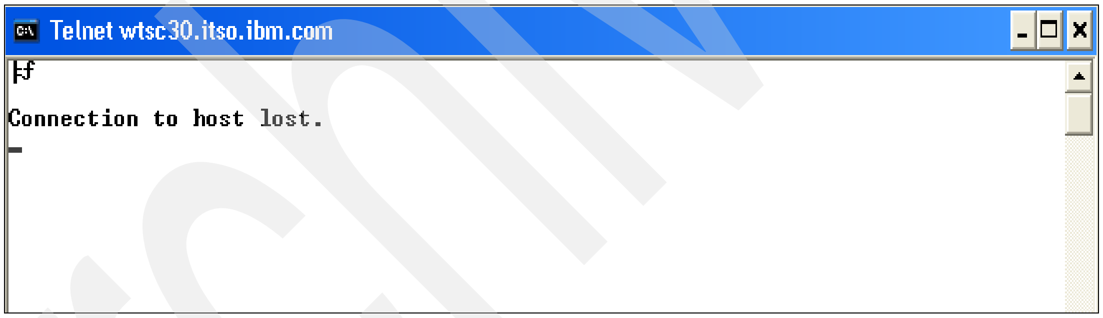


```
.<(+¦&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNO
<(+¦&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOP
(+¦&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQ
+¦&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR
¦&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\
&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\S
!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\ST
$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STU
*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUV
);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVW
;^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWX
^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXY
-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ
/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ0
,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ01
%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ012
_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ0123
>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ01234
?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ012345
```

Figure 6-5 Verification of the chargen service

Verifying 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.



```
ŀf

Connection to host lost.
_
```

Figure 6-6 Verification of the time service

Problem determination

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. Since INETD will not fork in debugging mode, the behavior of INETD may also change. Additionally, the -d option results in numerous messages sent to the STDERR data stream.

Refer to Chapter 9, “Remote execution” on page 257, and Chapter 8, “z/OS UNIX Telnet server” on page 247, for information regarding remote execution or otelnetd.

7

Mail servers

This chapter focuses on the mail services that are available in the z/OS V1R8 Communications Server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

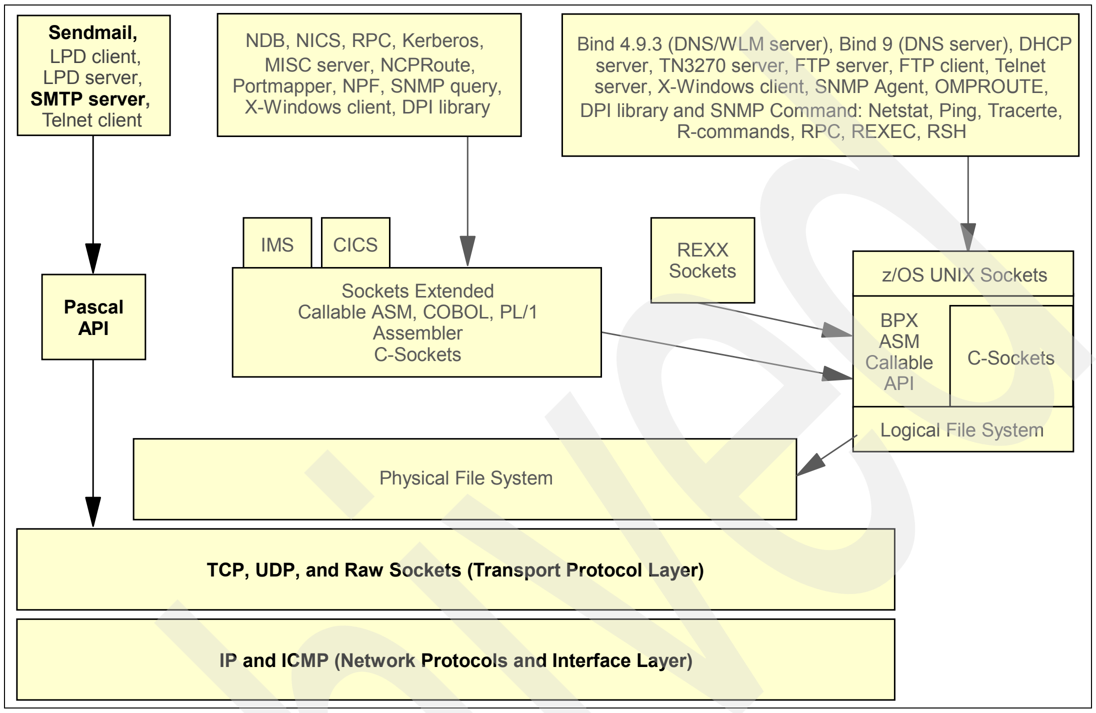
The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains comprehensive descriptions of the individual parameters for setting up mail services on z/OS. It also includes step-by-step checklists and supporting examples. It is not the intent of this book to duplicate the information in the guide, but to complement it with practical scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 7.1.2, "Additional information sources for mail servers" on page 220.

This chapter discusses the following.

Section	Topic
7.1, "Overview" on page 218	Discusses the basic concepts of mail servers
7.2, "Why the mail servers are important" on page 220	Discusses key characteristics of mail servers and why mail may be important in your environment
7.3, "The common design scenarios for the z/OS mail servers" on page 220	Presents commonly implemented mail server design scenarios, their dependencies, advantages, considerations, and our recommendations

7.1 Overview

As illustrated in Figure 7-1, SMTP and sendmail are standard applications provided with the z/OS Communications Server.



Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270 server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
REXX Sockets
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
z/OS UNIX Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

Figure 7-1 z/OS mail services

This section provides an overview of the TCP/IP application protocols dealing with electronic mail. Two mail functions are supported on z/OS:

- Simple Mail Transport Protocol (SMTP) in the MVS environment
- Sendmail as a z/OS UNIX component

The various standards that define the protocols for sending electronic mail are numerous. The Request For Comment (RFC) numbers that represent these standards and their terminology can be very confusing. We provide an overview of the main RFC numbers and their descriptions here.

RFC 821 defines a client/server protocol. As usual, the client SMTP is the one that initiates the session (that is, the sending SMTP), and the server is the one that responds (the receiving SMTP) to the session request. However, 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.

7.1.1 Basic concepts of mail servers

The basic Internet mail protocols provide mail (note) and message exchange between TCP/IP hosts. Facilities have been added for the transmission of data that cannot be represented as 7-bit ASCII text. There are three standard protocols that apply to mail of this kind. Each is recommended. The term Simple Mail Transfer Protocol (SMTP) is frequently used to refer to the combined set of protocols because they are so closely inter-related but, strictly speaking, SMTP is just one of the three. Normally, it is evident from the context which of the three protocols is being referenced. Whenever some doubt might exist, we refer to the STD or RFC numbers to avoid ambiguity.

The relationship of the z/OS SMTP server and the client network is illustrated in Figure 7-2.

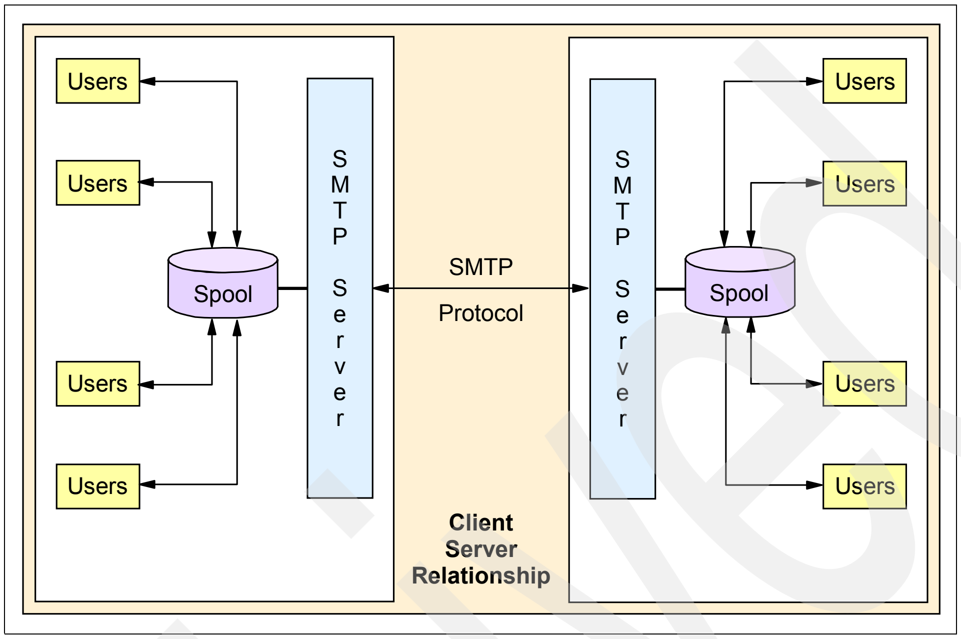


Figure 7-2 SMTP client/server relationship

Some of the most used standards are:

- 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 via 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 a set of document types other than plain text ASCII can be used in the mail body (the documents themselves are 7-bit ASCII containing 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, 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 specifies 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.

The list of mail RFC standards is much longer, but is out of scope for this book. For complete details on the standards and how they relate to each other, refer to *TCP/IP Tutorial and Technical Overview*, GG24-3376.

7.1.2 Additional information sources for mail servers

Detailed information about the mail servers and protocols can be found in the following documents:

- *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.

7.2 Why the mail servers are important

Electronic mail (e-mail) is probably the most widely used TCP/IP application. For most people, it has become an integral part of everyday life.

7.3 The common design scenarios for the z/OS mail servers

Mail services on the z/OS platform can be implemented in the z/OS MVS environment and in the z/OS UNIX environment. These two environments are discussed in the following sections.

7.3.1 z/OS SMTP server for an IP network, an NJE network, and relay servers

The z/OS SMTP server can be set up to distribute mail on an IP network. It can also perform the role of a gateway server transferring mail between the IP network and a local NJE network. It can also be configured to send unresolved non-local mail or all non-local mail to a Mail Transfer Agent (MTA) or relay server.

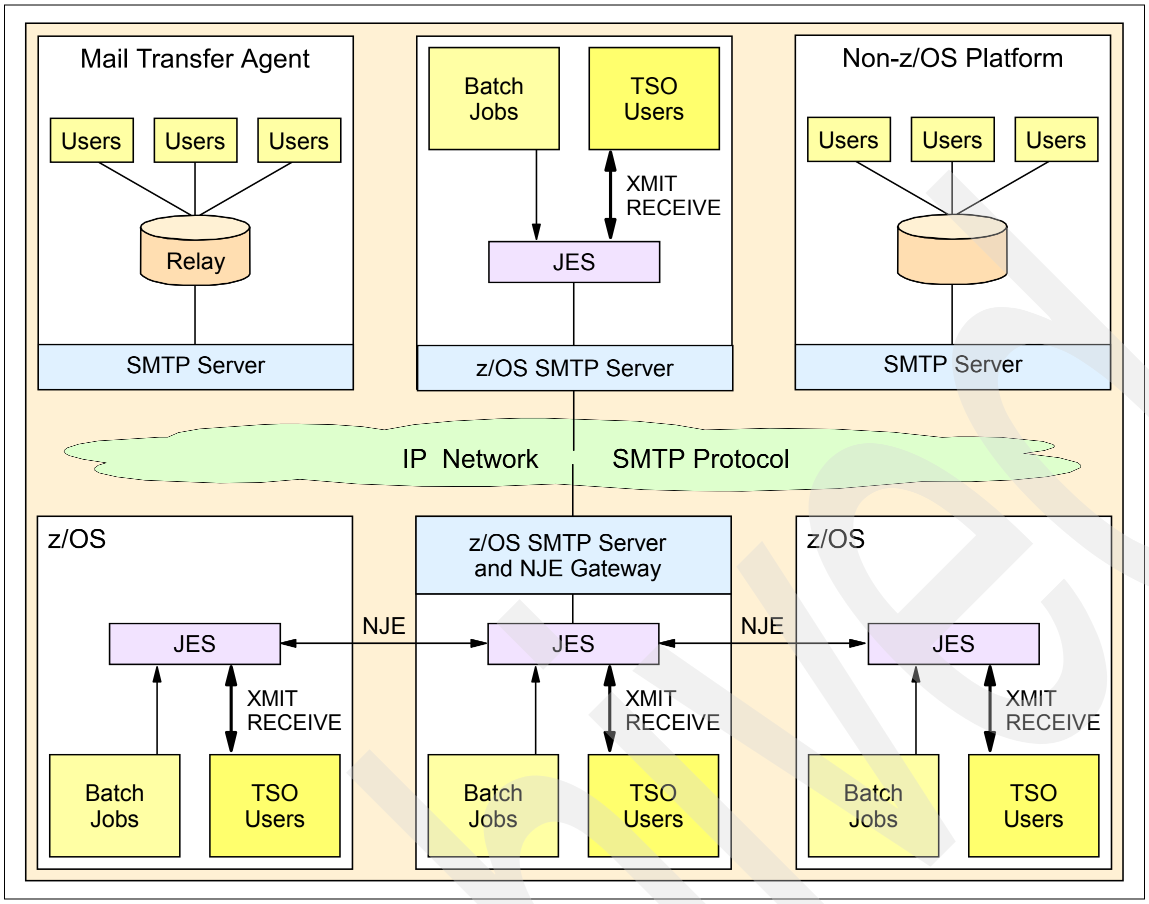


Figure 7-3 SMTP mail server: IP, NJE, relay relationships

We discuss these topics in the following sections:

- Description of z/OS SMTP server functions
- Dependencies of the z/OS SMTP server
- Advantages of using the z/OS SMTP server
- Considerations for using the z/OS SMTP server

Description of z/OS SMTP server functions

The functions are:

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 may 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

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 STMP 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 once 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.

Advantages of using the 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 may 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.

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.

Once 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 Sendmail and popper

In this section we discuss sendmail and popper.

Description

Sendmail is an industry standard mail application that is widely used on the Internet. The sendmail application included with the z/OS V1R7.0 Communications Server is based on sendmail version 8.12.1. Popper is a Post Office Protocol 3 (POP3) mail delivery server. It allows a remote user to use a remote mail user agent (MUA) to read, compose, and manage e-mail that has been delivered to z/OS.

Note: Due to the complexity of sendmail, we highly recommend that you become familiar with the industry-accepted publication about sendmail: *sendmail 3rd Edition* from O'Reilly & Associates, Inc. ISBN 1-56592-839-3. This book is well known as the *bat book*, because of the image of a bat on the cover.

MTA, MUA, and MDA

When discussing sendmail and popper, other books refer to Mail Transfer Agents (MTA), Mail User Agents (MUA), and Mail Delivery Agents (MDA). Let us take a moment to define these acronyms:

- Mail Transfer Agent (MTA)

  This is any application that sends a prepared e-mail message to a remote MTA. Sendmail, Microsoft Exchange, and Lotus® Domino® are examples of MTAs.

- Mail User Agent (MUA)

  This is any application that allows a user to read, compose, and manage e-mail. Lotus Notes®, Microsoft Outlook®, and Netscape Navigator are all applications that have MUA capabilities.

- Mail Delivery Agent (MDA)

  This is any application that sits between a MUA and MTU and manages delivering a message from a MTA to an MUA. A MDA is an optional piece of an e-mail system. Popper is an example of an MDA.

In z/OS UNIX, sendmail is a MTA, **`/bin/mail`** is a command-line MUA, and popper is an MDA. In our example configuration, we use sendmail as a 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 may use the cf.m4 and sample.mc files to create your own sendmail.cf output file or you may 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. In order 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 may view the statistics file using the commands **`mailstats -C </etc/sendmail.cf>`** or **`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 **`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 may find it empty, which implies that all messages have been sent. Or you may 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. The **`obrowse`** command may be used to read these files. Superuser permission is normally needed.

The following is a description of the 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.
10. For security reasons, this is the real recipient of the message.
11. Recipient's address.
12. Header information for the return path in case the message cannot be delivered.
13. HReceived: where the message came from.
14. Header information when the message was received by the MTA.
15. Header information about the sender.
16. 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:

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 very complex. If you browse through the configuration file found in /usr/lpp/tcpip/samples/sendmail/cf/sample.cf you will find very a complex example. In order 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 will 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 7.4.2, “Sendmail and popper” on page 235.

Dependencies

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 may list other directories and files that must be created before sendmail can start. In order 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.

Advantages

Sendmail and popper are full-featured industry standard mail applications. Sendmail is also capable of acting as a client that can be used to send e-mail. Sendmail supports MIME attachments and also has built-in security features such as TLS/SSL sockets.

Considerations

Sendmail can be very complex. Sendmail is incapable of interfacing with JES. If you want to submit batch job e-mail, then you should use SMTP rather than sendmail.

7.3.3 Recommendations

Depending on the electronic mail requirements for your z/OS mainframe, you can choose one or both of the following configurations:

- The z/OS SMTP server that uses the TCP/IP network, the NJE network, and relay servers
- Sendmail and popper

7.4 How the mail servers are implemented

We show the following implementations in this section:

- z/OS SMTP server for a TCP/IP network, NJE network, and relay servers
- Sendmail and popper

7.4.1 z/OS SMTP server for a TCP/IP network, NJE network, and relay servers

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 following topics are discussed below:

- "Implementation tasks for the z/OS SMTP server" on page 227
- "Configuration examples for the z/OS SMTP server" on page 232
- "Verification of the z/OS SMTP server" on page 234
- "Problem determination tasks for the z/OS SMTP server" on page 235

Implementation tasks for the z/OS SMTP server

The steps to configure the z/OS SMTP server are discussed here. Samples of each step are listed in "Configuration examples for the z/OS SMTP server" on page 232.

Update the TCPIP 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.

Update the security server, such as 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, refer to 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). Customize data set names to meet standards of your installation. There are some follow-on steps below that may require 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 TCPIP profile configuration data set.

Create the SMTP server configuration data set

A sample SMTP configuration member is in hlq.SEZAINST(SMTPCONF). Use this to set up a standard SMTP server. Once you are comfortable with the basic definitions, you can add optional functions such as the NJE gateway and relay server statements. For statement syntax, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For a discussion of statement use, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

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 STMP 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 will be used, or the batch method, or both, values for the following items must be establish 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. Once 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 may 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 timezone from the system Communication Vector Table (CVT) control block.

> **Note:** The value of SYSTZ is available in z/OS V1R8 and higher release.

ATSIGN This specifies a single-byte representation of the at symbol (@) for foreign languages.

For use of the SMTPNOTE clist, see *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780.

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, refer to *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

Determine whether NJE Gateway support is necessary: (NJE)

In order to implement the SMTP server as an NJE Gateway, some terminology needs to be discussed. Refer to the diagram in Figure 7-4.

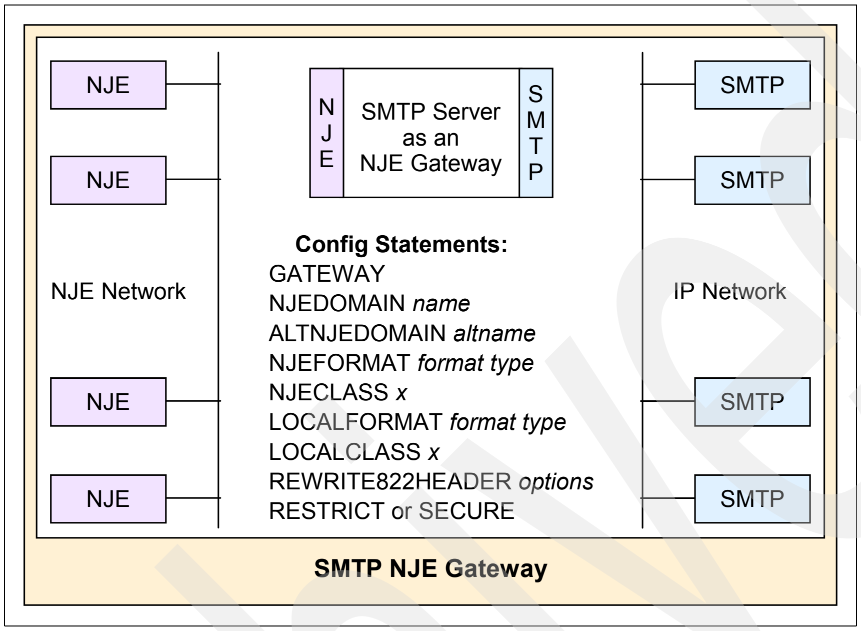


Figure 7-4 SMTP server as an NJE gateway server

If you intend to use the SMTP server as a gateway to the NJE network, these tasks must be considered:

- 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.

- 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. In order 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. The command scans the member for node definitions and builds the required file.

- 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
  ```

- Do 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
  ```

  And 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:

  ```
  mailfiledsprefix.SECURITY.MEMO with LRECL=255 and RECFM=VB
  ```

  It is dynamically allocated by the SMTP server when needed. You do *not* add a DD statement to the SMTP server procedure JCL.

  For examples and details on the format and syntax of the records for both data sets, you must refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

- Do 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 on the syntax of the RESTRICT statement you must refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

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 a Mail Transfer Agent (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 may 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.

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 on implementing and managing the exit, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Configuration examples for the z/OS SMTP server

Our SMTP server procedure is shown in Figure 7-5.

```
//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=TCPIPB.SMTPNJE.HOSTINFO,DISP=SHR
//CONFIG   DD DSN=TCPIPB.TCPPARMS(SMTPB&SYSCLONE.),DISP=SHR
//*SECTABLE DD DSN=TCPIPB.SMTP.SECTABLE,DISP=SHR
//*SMTPRULE DD DSN=TCPIPB.SMTP.RULE,DISP=SHR
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.),DISP=SHR
```

Figure 7-5 SMTP server Proc JCL

Portions of our SMTP server configuration data set are shown in the following figures:

- Figure 7-6 on page 233 shows unique parameters for this server instance.
- Figure 7-7 on page 234 shows NJE-related parameters.
- Figure 7-8 on page 234 shows relay server related parameters.
- Figure 7-9 on page 234 shows restrict or secure related settings.

```
;*********************************************************************
;  Defaults that normally aren't changed:
;*********************************************************************
PORT 25                  ; Port to accept incoming mail
BADSPOOLFILEID TCPMAINT ; Mailbox where unreadable spool files and
                         ; looping mail are transferred.
LOG                      ; Log all SMTP mail delivered
INACTIVE 180             ; Time-out for inactive connections
FINISHOPEN 120           ; Time-out for opening TCP connections
RETRYAGE 3               ; Keep retrying mail delivery for 3 days
WARNINGAGE 1             ; Warn sender that mail has been undeliverable
                         ; for 1 day, but that attempts to deliver the
                         ; mail will continue for another 2 days.
RETRYINT 20              ; Retry mail delivery every 20 minutes
MAXMAILBYTES 524288      ; Largest mail to accept over a TCP connection
RESOLVERRETRYINT 20      ; Retry pending name resolutions every 20
                         ; minutes
RCPTRESPONSEDELAY 60     ; How long to delay RCPT TO: response when
                         ; waiting for an address resolution.
TEMPERRORRETRIES 0       ; How many times to retry temporary delivery
                         ; errors.  The default, 0, means retry for
                         ; RETRYAGE days; otherwise the mail is returned
                         ; after this number of deliver attempts.
SPOOLPOLLINTERVAL 30     ; Amount of time in seconds between spool
                         ; polling
TIMEZONE SYSTZ           ; Specifies the printable letter name of
                         ; the local time zone.  Maximum number of chars
                         ; is 5. If the value of parameter is SYSTZ then
                         ; the time zone is gotten from the MVS system
                         ; using the value in the CVT variable CVTLDTO.
                         ; It is represented as a + or - offset from
                         ; the local time for the region (ie -0400).
                         ; If SYSTZ is not configured, remember to change
                         ; this for daylight saving time if appropriate
                         ; for your region.
; DEBUG                  ; Normally not used, causes debug information to
                         ; be written to the SMTP DEBUG file.
;*********************************************************************
;  Installation specific statements
;*********************************************************************
ALTTCPHOSTNAME SC30MAIL
MAILFILEDSPREFIX TCPIPB ; Data set prefix name for queued mail files
MAILFILEUNIT SYSDA       ; MVS unit name for new file allocations
; MAILFILEVOLUME volume ; MVS volume name for new file allocations
POSTMASTER CS06 CS02
SMSGAUTHLIST
    CS06
    CS02
ENDSMSGAUTHLIST
; OUTBOUNDOPENLIMIT 30
```

Figure 7-6   SMTP server configuration data set: unique parameters

```
;**********************************************************************
; 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 SCTESTNET ; user02%SCNJENET@SC30MAIL.raleigh.ibm.com
NJEFORMAT PUNCH        ; NJE recipients receive mail in punch format
NJECLASS B             ; 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 B           ; Spool class for local mail delivered by SMTP
REWRITE822HEADER YES NOPRINT
```

Figure 7-7 SMTP server configuration data set: NJE parameters

```
;**********************************************************************
;      Send ALL non-local mail to an MTA relay server
;**********************************************************************
IPMAILERADDRESS 9.22.114.229 ; Relay Server
RESOLVERUSAGE NO      ; Send all non-local mail to Relay Server
```

Figure 7-8 SMTP server configuration data set: relay server parameters

```
Note that RESTRICT and SECURE are mutually exclusive
;**********************************************************************
RESTRICT RETURN          ; Return mail from restricted users
  cs01@us.ibm.com        ; Don't accept any mail from cs01
  cs01@SCNJENET          ; via NJE or TCP network.
  cs01@*                 ;This line takes the place of previous 2 lines
 *@badsite               ;Don't accept mail from anyone at badsite
ENDRESTRICT
;
; SECURE
```

Figure 7-9 SMTP server configuration data set: restrict and secure parameters

Note: RESTRICT and SECURE are both optional settings. However, they are also mutually exclusive.

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 use a SYSTCPT DD statement in the SMTP cataloged procedure.

Note: The SMSG command works when issued from TSO 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.

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 7-1 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 7-1 SMTP Spool file

```
Received: from SC32 by WTSC32B.ITSO.IBM.COM (IBM MVS SMTP CS V1R8)
   with BSMTP id JOB00757; Thu, 31 Aug 06 11:33:17 -0400 1
Date: Thu, 31 Aug 06 11:33:17 -0400 1
From: <CS03@SC32MAIL.ITSO.IBM.COM>

THIS IS A TEST MAIL.
```

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 as described in the *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780. 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, by using the SMSG SMTP command. Some of the commands are:

- SMSG SMTP HELP
- SMSG SMTP QUEUES
- SMSG SMTP STATS
- SMSG SMTP DEBUG or NODEBUG
- SMSG SMTP TRACE or NOTRACE
- SMSG SMTP STARTEXIT or STOPEXIT
- SMSG SMTP EXPIRE IPaddress
- SMSG SMTP SHUTDOWN

These administrative tasks are discussed in more detail in the *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780.

7.4.2 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 e-mail from z/OS.

Implementation tasks

The following steps must be performed to start sendmail.

Important: The following steps must be performed by the superuser ID, uid 0. If you use another user ID sendmail may issue the following message:

```
EZZ9927I Permission denied (real uid not trusted).
```

1. Create the /etc/mail directory.
2. Copy the sample configuration.
3. Create the queue directory.
4. Create the local-host-names file.
5. Create the popper maildrop directory.
6. Create the alias file for local recipients.
7. Update /etc/inetd.conf for popper.

8. Update /etc/services.
9. Update PROFILE.TCPIP.
10. Start sendmail.
11. Start inetd.

Create the /etc/mail directory

From the z/OS UNIX System Services shell issue:

```
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:

```
chmod 755 /etc
chmod 755 /etc/mail
```

Copy the sample configuration file

From the z/OS UNIX System Services shell issue:

```
cp /usr/lpp/tcpip/samples/sendmail/cf/sample.cf /etc/mail/sendmail.cf
```

Create the queue directory

From the z/OS UNIX System Services shell issue:

```
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 for sendmail to start.

From the z/OS UNIX System Services shell issue:

```
touch /etc/mail/local-host-names
```

Create the popper maildrop directory

From the z/OS UNIX System Services shell issue:

```
mkdir /usr/mail/popper
chmod 777 /usr/mail/popper
```

Create the alias file

The alias file maps names in e-mail addresses to local user accounts. You must list in the alias file all local users who are to receive e-mail. Our alias file is shown in Example 7-2.

Example 7-2 Contents of our /etc/mail/aliases file

```
MAILER-DAEMON:IBMUSER
postmaster:IBMUSER
cs02: CS02
cs03: CS03
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.

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 7-3. If you have already started INETD as described in Chapter 6, “INETD” on page 207, then you have already completed this step.

Example 7-3 /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 7-4. If you have already started INETD as described in Chapter 6, “INETD” on page 207, then you have already completed this step.

Example 7-4 /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. Since popper uses port 110 TCP, and is started by INETD, we need to reserve port 110 to INETD. Example 7-5 shows our PROFILE.TCPIP statements.

Example 7-5 PORT statements in PROFILE.TCPIP

```
25    TCP     OMVS
110   TCP     INETD1
```

Start sendmail

We use the following commands at the z/OS UNIX shell to start sendmail:

```
export _BPX_JOBNAME=SENDMAIL
/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.

Start inetd

If you have already started INETD as described in Chapter 6, “INETD” on page 207, then you have already completed this step. Otherwise, from the z/OS UNIX System Services shell issue:

```
export _BPX_JOBNAME=INETD
/usr/sbin/inetd &
```

Verification

As shown in Example 7-6, we can use the **`ps`** command to verify that sendmail has started.

Example 7-6 Output of ps command

```
83886093 ?         0:00 /usr/sbin/sendmail
```

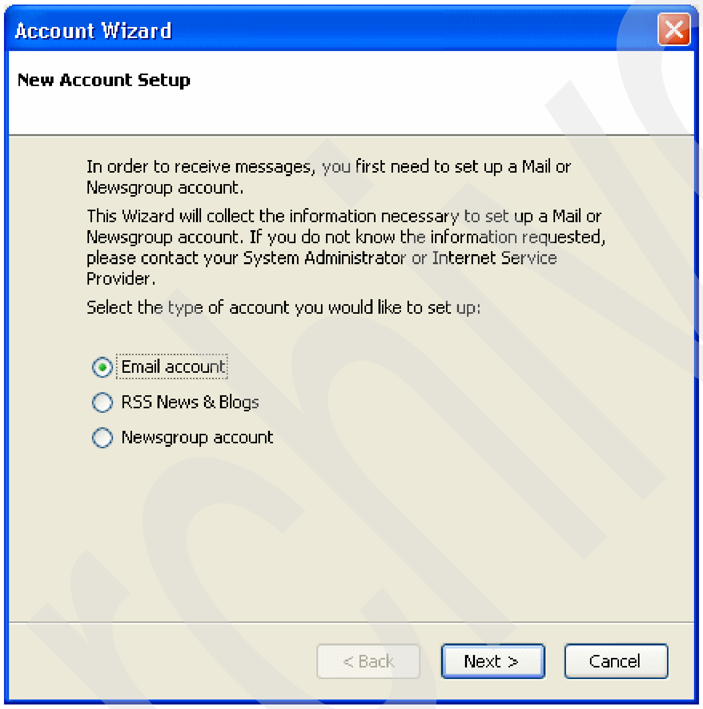
Next, as shown in Example 7-7 on page 238, we can use **`netstat`** to determine whether sendmail has put a listener up on port 25 TCP.

Example 7-7 Output of netstat

```
SENDMAIL 0000003E Listen
  Local Socket:   0.0.0.0..25
  Foreign Socket: 0.0.0.0..0
```

Finally, we use the freely available e-mail client Mozilla Thunderbird as a MUA to check our e-mail. We obtained Thunderbird 1.7 from http://www.mozilla.com and installed it on a PC. While installing Thunderbird we were prompted to set up an e-mail account. The following five screen captures show the account settings needed to allow Thunderbird to access sendmail and popper running on our 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 e-mail account, as shown in Figure 7-10.


Account Wizard

New Account Setup

In order to receive messages, you first need to set up a Mail or Newsgroup account.

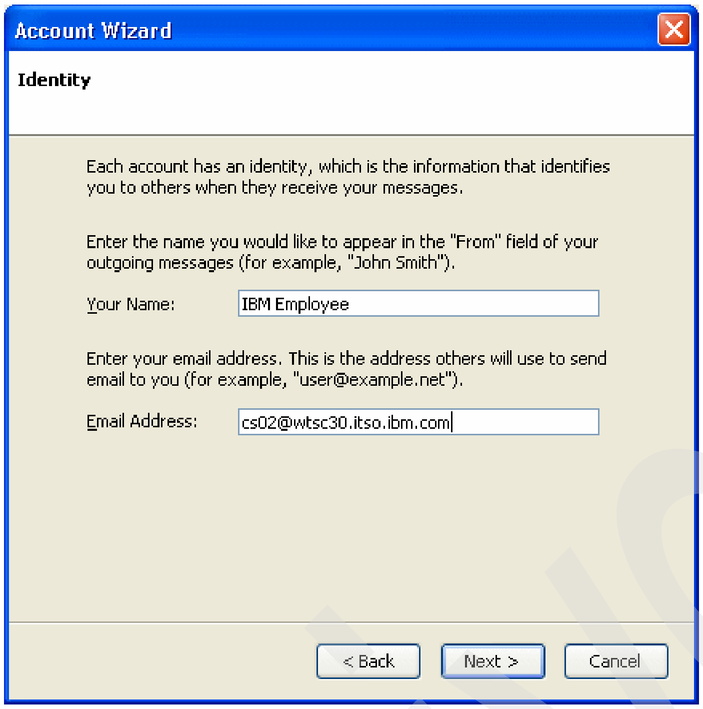
This Wizard will collect the information necessary to set up a Mail or Newsgroup account. If you do not know the information requested, please contact your System Administrator or Internet Service Provider.

Select the type of account you would like to set up:

[✓ Email account]
RSS News & Blogs
Newsgroup account

Figure 7-10 First panel of the Mozilla Thunderbird account wizard

The second panel in the account wizard asks for your name and e-mail address, as seen in Figure 7-11 on page 239.



Each account has an identity, which is the information that identifies you to others when they receive your messages.

Enter the name you would like to appear in the "From" field of your outgoing messages (for example, "John Smith").

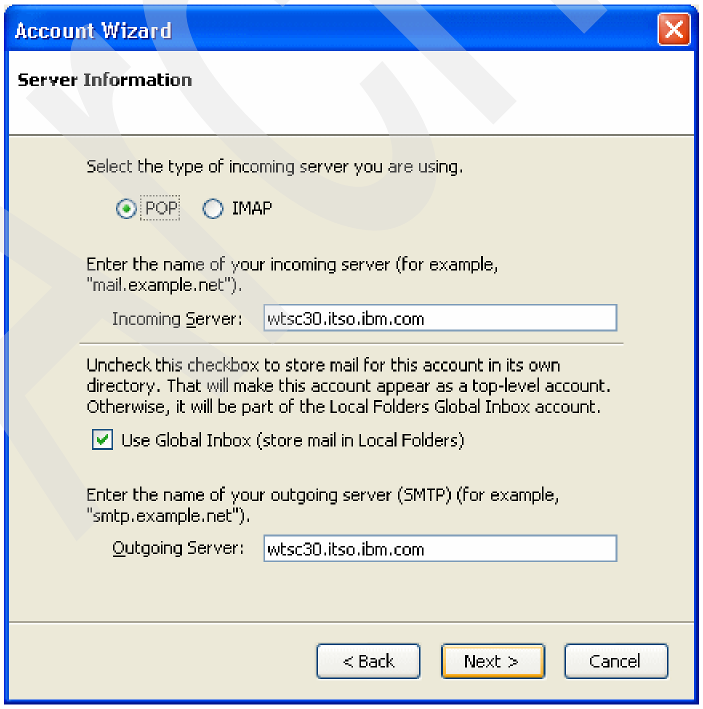
Your Name: IBM Employee

Enter your email address. This is the address others will use to send email to you (for example, "user@example.net").

Email Address: cs02@wtsc30.itso.ibm.com

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 since we are running popper and sendmail on the same LPAR. Our settings for the third panel are shown in Figure 7-12.



Select the type of incoming server you are using.

[✓ POP] IMAP

Enter the name of your incoming server (for example, "mail.example.net").

Incoming Server: wtsc30.itso.ibm.com

Uncheck this checkbox to store mail for this account in its own directory. That will make this account appear as a top-level account. Otherwise, it will be part of the Local Folders Global Inbox account.

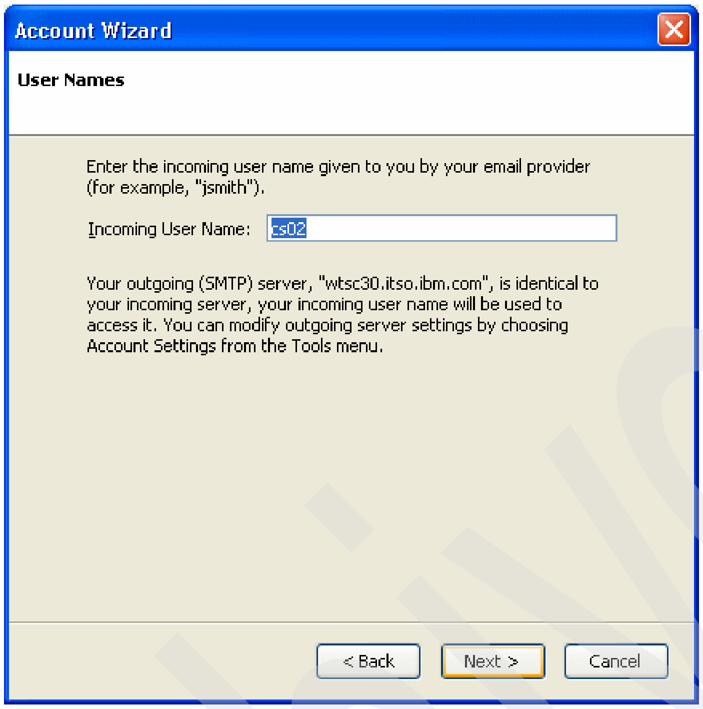
[✓] Use Global Inbox (store mail in Local Folders)

Enter the name of your outgoing server (SMTP) (for example, "smtp.example.net").

Outgoing Server: wtsc30.itso.ibm.com

Figure 7-12   Third screen of the Mozilla Thunderbird account wizard

The fourth screen in the account wizard asks for the user ID to present to the incoming and outgoing servers. Again, since 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.



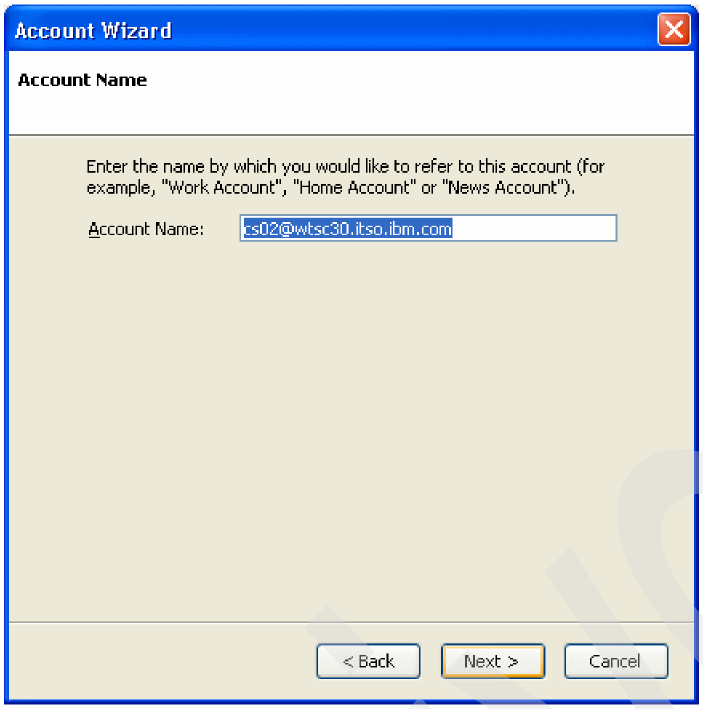
Enter the incoming user name given to you by your email provider (for example, "jsmith").

Incoming User Name: cs02

Your outgoing (SMTP) server, "wtsc30.itso.ibm.com", is identical to your incoming server, your incoming user name will be used to access it. You can modify outgoing server settings by choosing Account Settings from the Tools menu.

Figure 7-13 Fourth screen of the Mozilla Thunderbird account wizard

The fifth screen of the wizard asks for you to name the account. We used the name shown in Figure 7-14 on page 241.

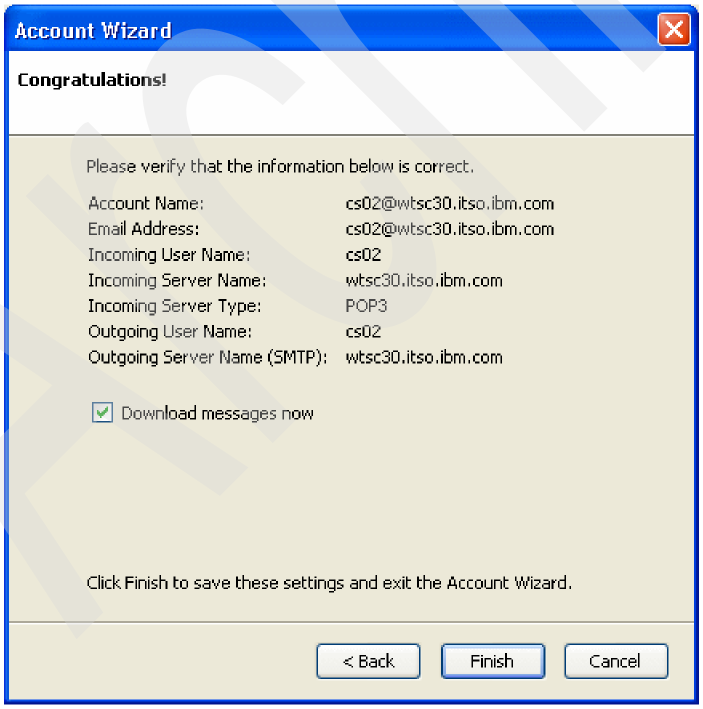


Enter the name by which you would like to refer to this account (for example, "Work Account", "Home Account" or "News Account").

Account Name: cs02@wtsc30.itso.ibm.com

Figure 7-14 Fifth screen 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 screen is shown in Figure 7-15.



Please verify that the information below is correct.

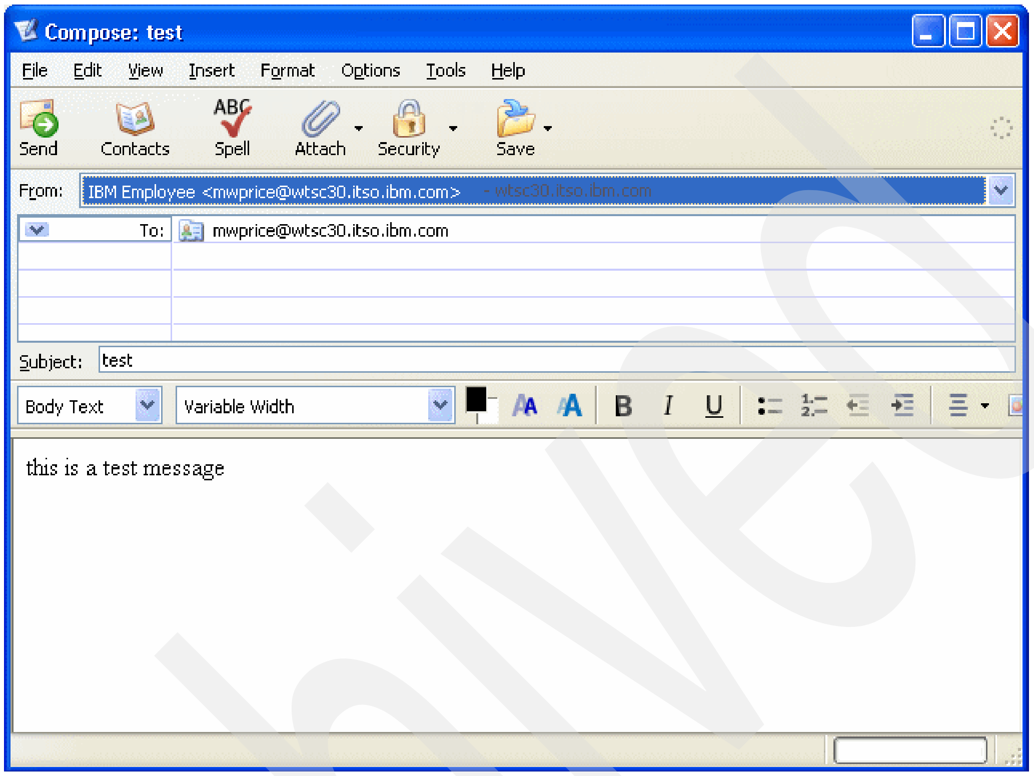
Account Name:	cs02@wtsc30.itso.ibm.com
Email Address:	cs02@wtsc30.itso.ibm.com
Incoming User Name:	cs02
Incoming Server Name:	wtsc30.itso.ibm.com
Incoming Server Type:	POP3
Outgoing User Name:	cs02
Outgoing Server Name (SMTP):	wtsc30.itso.ibm.com

✓ Download messages now

Click Finish to save these settings and exit the Account Wizard.

Figure 7-15 Final panel of the Mozilla Thunderbird account wizard

Now that Mozilla Thunderbird is configured to communicate with our system running sendmail, we send a test message to ourself to test the environment. Figure 7-16 shows the message we sent.


Compose: test
File Edit View Insert Format Options Tools Help
Send Contacts Spell Attach Security Save

From: IBM Employee <mwprice@wtsc30.itso.ibm.com> - wtsc30.itso.ibm.com

To: mwprice@wtsc30.itso.ibm.com

Subject: test

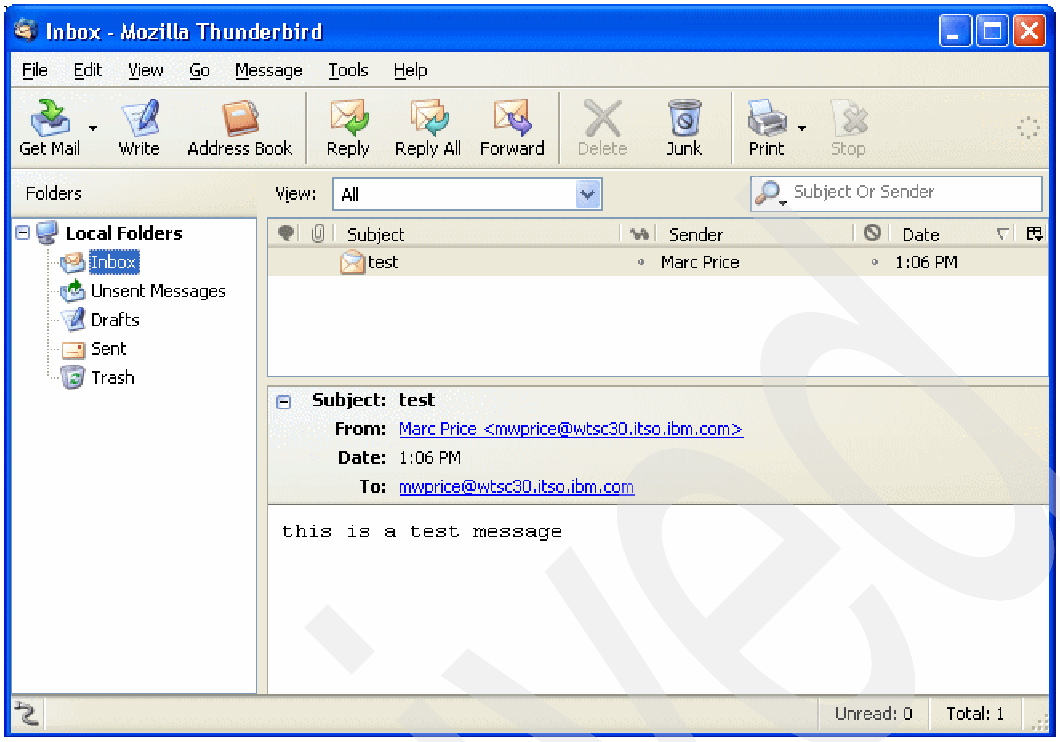
Body Text, Variable Width

this is a test message

Figure 7-16 Outgoing test message

After a brief wait of approximately 10 seconds, Thunderbird confirmed that the e-mail 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 few minute wait, we clicked the **Get Mail** button in Thunderbird and received the message shown in Figure 7-17 on page 243.


Inbox - Mozilla Thunderbird
File Edit View Go Message Tools Help
Get Mail Write Address Book Reply Reply All Forward Delete Junk Print Stop
Folders
View: All
Subject Or Sender
Local Folders
Inbox
Unsent Messages
Drafts
Sent
Trash
Unread: 0
Total: 1

Subject	Sender	Date
test	Marc Price	1:06 PM

Subject: test
From: Marc Price <mwprice@wtsc30.itso.ibm.com>
Date: 1:06 PM
To: mwprice@wtsc30.itso.ibm.com

```
this is a test message
```

Figure 7-17 Inbound message

We have now shown that both sendmail and popper are up and running and able to deliver mail locally.

Problem determination

The sendmail bat book documents the detailed internal traces available in sendmail. Refer to *sendmail 3rd Edition* from O'Reilly & Associates, Inc. ISBN 1-56592-839-3.

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.

7.4.3 Sendmail as a client

In this section we discuss the use of sendmail as an e-mail client. We compose a message on z/OS and use sendmail to send it to another e-mail address. We also discuss how to create an e-mail 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 FTPed the image file 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

Implementation tasks

The tasks are:

1. Set up sendmail and popper.
2. Compose an RFC 822 compliant message.
3. Encode our attachment.
4. Join the encoded file with the text message.

5. Add MIME attachment headers to our message.
6. Invoke sendmail in the client mode to send the message.

Set up sendmail and popper

First, set up sendmail and popper as described in 7.4.2, “Sendmail and popper” on page 235. The configuration file, alias file, and local-host-names files are all required for sendmail regardless of whether sendmail is run in server 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 e-mail message. You can read 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 7-8.

Example 7-8 RFC 822 compliant message in /tmp/test-msg

```
To: cs02@wtsc30.itso.ibm.com
From: cs02@wtsc30.itso.ibm.com
Date: Thu Dec 29 00:39:28 2006
Subject: test message

Hello,
   This is a test message using sendmail.

MWP
```

Encode the attachment

In order to send binary data via e-mail, you must encode the binary data into an e-mail safe format. Modern e-mail applications use an encoding called base64 to encode binary files. We found the easiest way to perform base64 encoding was to download base64 encoding/decoding software from the Internet. We used the b64 program available at http://base64.sourceforge.net/. After downloading the source, we compiled the b64.c program from the z/OS UNIX shell using the command **`cc -o b64 b64.c`**. 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 7-9 shows the contents of our base64 encoded file.

Example 7-9 base64 encoded image in /tmp/ibm.gif

```
R0lGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR^M
zIuz3nzHtHyGVTOhYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4^M
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADs=^M
```

Join the encoded file with the text message

We can 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 7-10.

Example 7-10 /tmp/test-message after joining the text and binary files

```
To: cs02@wtsc30.itso.ibm.com
From: cs02@wtsc30.itso.ibm.com
```

```
Date: Thu Dec 29 00:39:28 2006
Subject: test message

Hello,
   This is a test message using sendmail.

MWP

R0lGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR^M
zIuz3nzHtHyGVTOhYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4^M
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADs=^M
```

Add MIME attachment headers to the message

Multipurpose Internet Mail Extensions (MIME) are additional headers that can be included in e-mail messages. In e-mail messages, an attachment is simply binary data surrounded by MIME headers. A e-mail client capable of handling attachments simply looks for the MIME headers and separates the e-mail 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 7-11. RFC 2045 and RFC 2046 (available at `http://www.ietf.org/rfc/rfc2045.txt` and `http://www.ietf.org/rfc/rfc2046.txt`) describe MIME in full detail.

Example 7-11 /tmp/test-message with MIME headers

```
To: cs02@wtsc30.itso.ibm.com
From: cs02@wtsc30.itso.ibm.com
Date: Thu Dec 29 00:39:28 2006
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.

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"

R0lGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR
zIuz3nzHtHyGVTOhYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADs=
-M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y--
```

Important: We found that the b64.c program, as provided, added “^M” characters to the end of each line of base64 encoded text. We manually removed the “^M” characters. “^M” 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, which only use the line feed character to mark the end of a line.

The following line of code in the encode() routine of the b64.c program is responsible for printing both carriage return and line feed: fprintf(outfile, "\r\n");. If the line were changed to fprintf(outfile, "\n");, then b64 would not append the unnecessary “^M” characters.

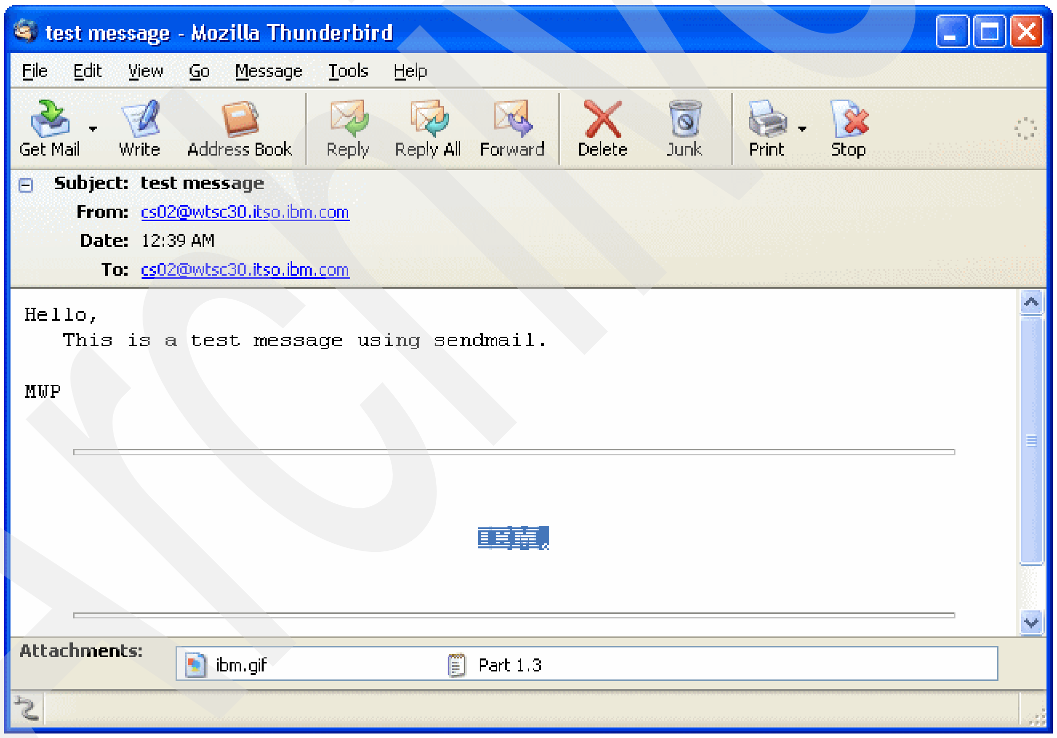
Invoke sendmail in the client mode to send the message

At last we are able to send our message. Invoke sendmail with:

```
/usr/sbin/sendmail -C /etc/mail/sendmail.cf -bm -t < /tmp/test-message
```

Verification

To verify that our attachment was successfully sent, we simply checked our e-mail with Mozilla Thunderbird, which we set up in 7.4.2, “Sendmail and popper” on page 235. After clicking **Get Mail** in Mozilla Thunderbird, we received the message shown in Figure 7-19.


test message - Mozilla Thunderbird
File Edit View Go Message Tools Help
Get Mail Write Address Book Reply Reply All Forward Delete Junk Print Stop

Subject: test message
From: cs02@wtsc30.itso.ibm.com
Date: 12:39 AM
To: cs02@wtsc30.itso.ibm.com

```
Hello,
   This is a test message using sendmail.

MWP
```

Attachments: ibm.gif Part 1.3

Figure 7-19 Message with attachment that was received from our z/OS system

Problem determination

The sendmail bat book documents the detailed internal traces available in sendmail. Refer to *sendmail 3rd Edition* from O'Reilly & Associates, Inc. ISBN 1-56592-839-3.

8

z/OS UNIX Telnet server

The z/OS UNIX Telnet server, also known as otelnetd, enables remote Telnet clients to log directly on to the z/OS UNIX System Services shell without having to log onto TSO. This chapter focuses on the z/OS UNIX Telnet server functions that are available in the z/OS V1R8 Communications server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains excellent descriptions of the individual parameters for setting up the z/OS UNIX Telnet server. It also includes step-by-step checklists and supporting examples. It is not the intent of this book to duplicate the information in the guide, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 8.1.2, “For additional information” on page 253.

This chapter uses the terms *otelnetd*, and *z/OS UNIX Telnet server* to refer to the Telnet server that provides access to z/OS UNIX, which is included with the z/OS V1R7.0 Communications Server. Other manuals may also refer to the same application as *oTelnetD*, or *the Telnet daemon*.

This chapter discusses the following.

Section	Topic
8.1, “Overview” on page 248	Discusses the basic concepts of the z/OS UNIX Telnet server
8.2, “Why the z/OS UNIX Telnet server is important” on page 253	Discusses key characteristics of the z/OS UNIX Telnet server and why it may be important in your environment
8.3, “The common design scenarios for z/OS UNIX Telnet server” on page 253	Presents commonly implemented z/OS UNIX Telnet server design scenarios, their dependencies, advantages, considerations, and our recommendations
8.4, “How the z/OS UNIX Telnet server is implemented” on page 254	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

8.1 Overview

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.

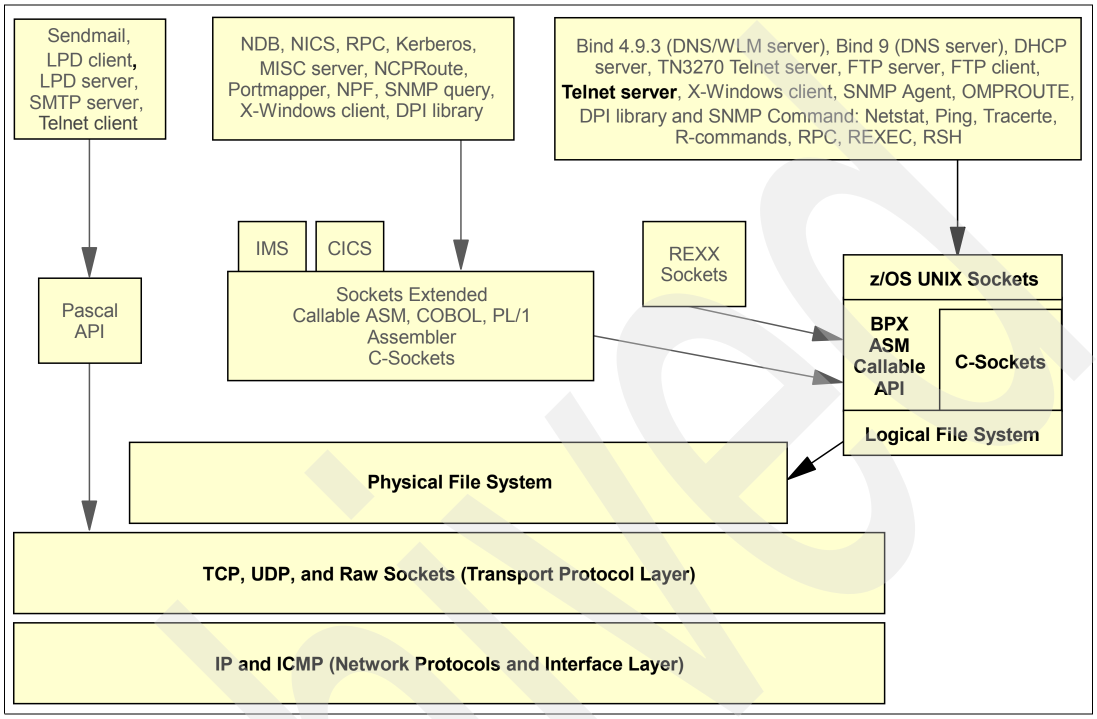

Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270 Telnet server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
REXX Sockets
z/OS UNIX Sockets
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

Figure 8-1 otelnetd application services

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 1, “TN3270E Telnet server” on page 1). The z/OS UNIX server can optionally use Kerberos 5 authentication and DES encryption.

8.1.1 Basic concepts

The z/OS UNIX Telnet server is started by INETD, which is discussed in Chapter 6, “INETD” on page 207. Once INETD has created a TCP connection with a client, INETD forks and executes an instance of otelnetd. The relationships between otelnetd and the TN3270E Telnet server and the relationship between otelnetd and INETD are shown in Figure 8-2 on page 249.

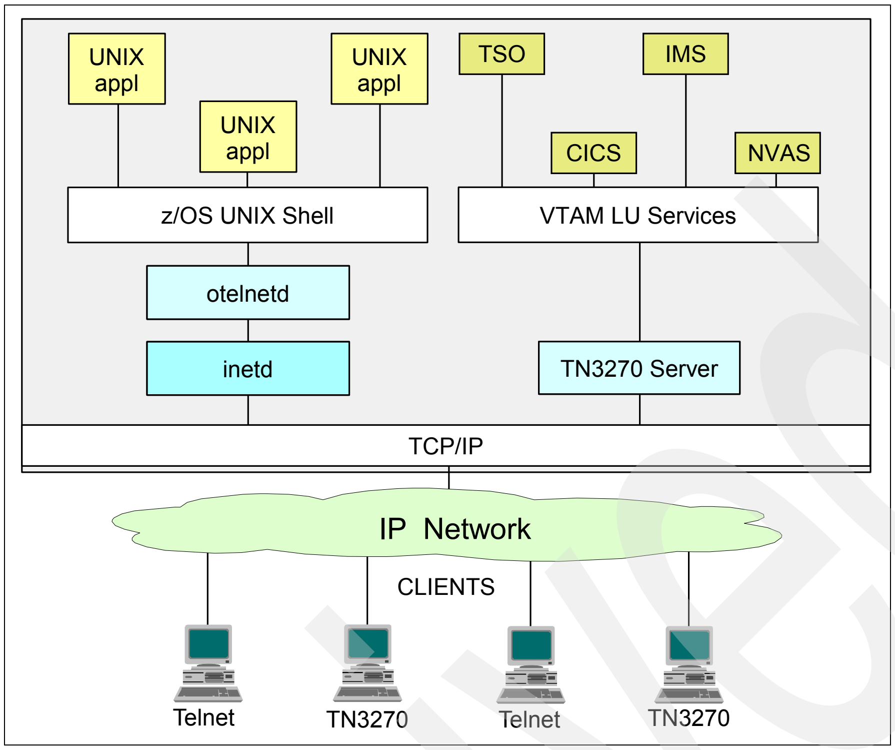


Figure 8-2 INETD and otelnet relationship

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 8-3.

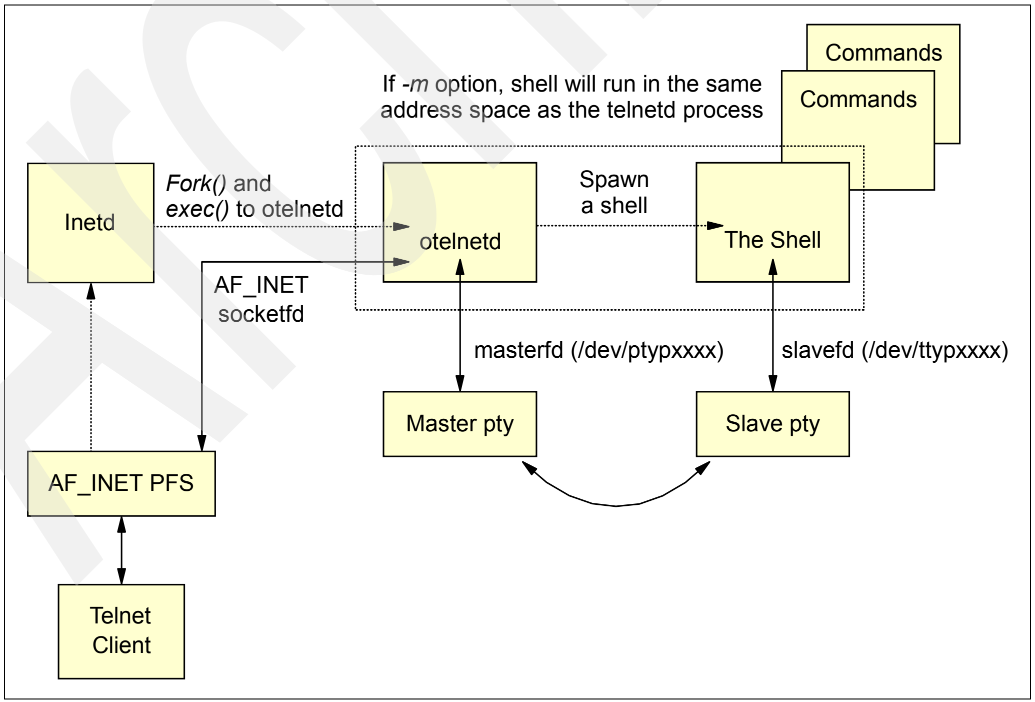


Figure 8-3 otelnetd interactions with INETD and z/OS UNIX

ASCII/EBCDIC translation

Since 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 via urgent data over the pseudo terminal connection (the master/slave pty interface). If a user selects to change her code page, the **chcp** command could be executed as part of the user's login process, for example, via the user's $HOME/.profile or $HOME/.setup shell scripts.

For more information about the **chcp** command, refer to *z/OS UNIX System Services Command Reference*, SA22-7802.

Banner pages

An environment-specific banner page can be created in the file /etc/banner (shown in Example 8-1) using standard edit functions. It will be displayed at the client workstation after the user ID and password have been entered.

Example 8-1 File containing the banner page

```
***********************************************************************
*                                                                     *
*  Welcome to z/OS V1R8 UNIX System Services                          *
*                                                                     *
*  This is the /etc/banner page for Telnet server use.                *
*                                                                     *
***********************************************************************
```

With the /etc/banner file above, the welcome page in Example 8-2 was displayed on our terminal.

Example 8-2 Telnet welcome page

```
***********************************************************************
*                                                                     *
*  Welcome to z/OS V1R8 UNIX System Services                          *
*                                                                     *
*  This is the /etc/banner page for Telnet server use.                *
*                                                                     *
***********************************************************************
IBM
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 2000
(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.

BTHOMPS:/u/bthomps: >
```

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/OZ 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 refer to *z/OS UNIX System Services Planning*, GA22-7800.

The terminal capabilities database

For telnet sessions, you may need some definitions of terminal capabilities in order 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 very 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, may 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 suffix .ti. 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 UNIX System Services 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

Directory	Terminal definition files
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.

In order 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 UNIX System Services 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. Refer to 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, refer to *Termcap & Terminfo*, published by O'Reilly & Associates.

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. Refer to *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926, for more information.

Starting both the z/OS UNIX Telnet server and the TN3270E Telnet server

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 a different IP address or host name.

8.1.2 For additional information

For additional information, refer to:

- *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.

8.2 Why the z/OS UNIX Telnet server is important

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 very useful if you have UNIX applications that run under z/OS UNIX.

8.3 The common design scenarios for z/OS UNIX Telnet server

The z/OS UNIX Telnet server is a simple application with limited configuration options. Therefore, we only show a single design scenario.

8.3.1 z/OS UNIX Telnet server

This section discusses otelnetd in a typical environment, with no authentication or encryption.

Dependencies

The z/OS Telnet server requires an active TCP/IP stack and INETD. We also highly recommended that you start syslogd to manage messages that may be generated by the otelnetd server.

Advantages

The z/OS UNIX Telnet server is easy to implement and provides a quick method to access the z/OS UNIX shell.

Considerations

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.

Note: The configuration we show may 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 *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342.

8.3.2 Recommendations

We recommend running the z/OS UNIX Telnet server only if you require direct access to the z/OS UNIX shell.

8.4 How the z/OS UNIX Telnet server is implemented

Based upon our recommendations from 8.3.2, “Recommendations” on page 254, we only show implementation details for the z/OS UNIX Telnet server.

8.4.1 z/OS UNIX Telnet server

In order 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.

Implementation tasks

The tasks are:

1. Reserve a PORT in PROFILE.TCPIP for otelnetd.
2. Create a port entry in ETC.SERVICES or /etc/services for otelnet.
3. Add a line to 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 207, then you have already completed the necessary configuration and otelnetd may already be active.

Reserving a PORT in PROFILE.TCPIP

While 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 TCP INETD1 BIND 10.20.10.251
```

Creating a port entry in /etc/services

An entry in /etc/services is required to map the otelnet service name to a port number. Our entry in /etc/services is shown in Example 8-4 and maps the service otelnet to port 23 tcp.

Example 8-4 Statement in /etc/services

```
otelnet        23/tcp
```

INETD configuration file changes for otelnetd

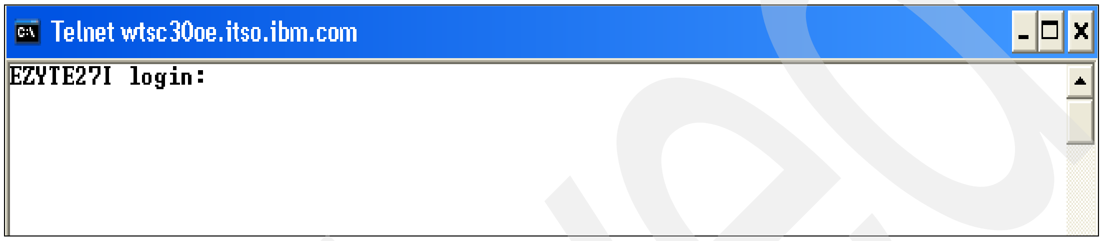
An entry is required in the INETD configuration file so that INETD will set up a listening socket for otelnetd. The application otelnetd can accept a number of command-line parameters that are specified in the INETD configuration file. All of the possible command-line parameters are discussed in detail 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. Our INETD configuration file entry is shown in Example 8-5.

Example 8-5 Statement in INETD configuration file

```
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
```

Verification

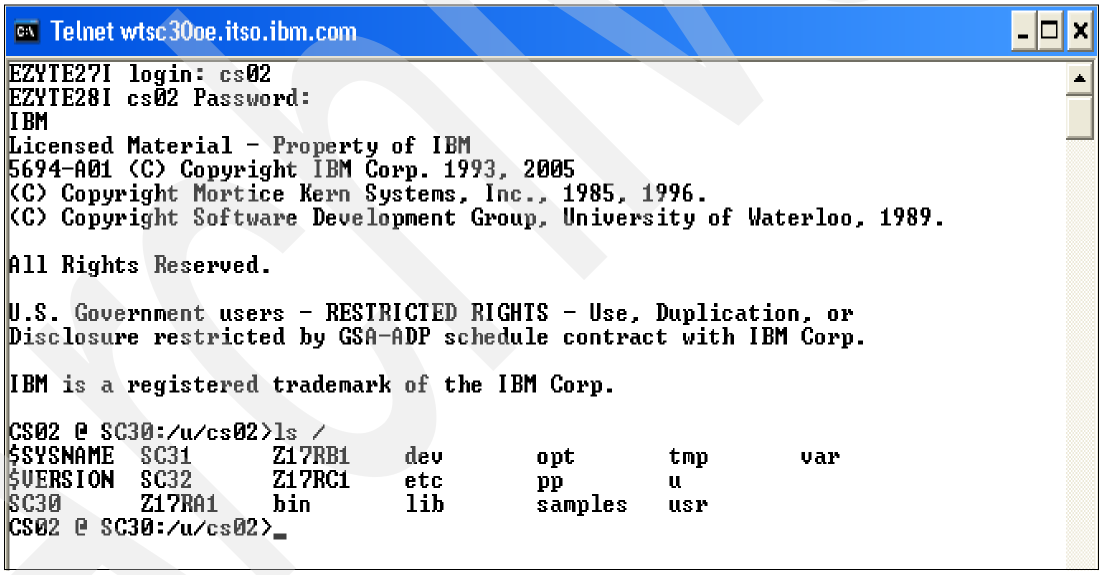
Verification of otelnetd is simple: use your favorite Telnet client, connect to the address and port that otelnetd is listening on, and log in. If you can log in and everything is readable, then otelnetd is working correctly. Figure 8-4 shows the otelnetd login screen.



```
EZYTE27I login:
```

Figure 8-4 otelnetd login screen

Figure 8-4 shows the output of a /bin/ls command issued after successful login.



```
EZYTE27I login: cs02
EZYTE28I cs02 Password:
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2005
(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.

CS02 @ SC30:/u/cs02>ls /
$SYSNAME  SC31      Z17RB1    dev       opt       tmp       var
$VERSION  SC32      Z17RC1    etc       pp        u
SC30      Z17RA1    bin       lib       samples   usr
CS02 @ SC30:/u/cs02>_
```

Figure 8-5 /bin/ls issued after successful login

Problem determination

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. The *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, provides a detailed example of the trace output generated by otelnetd.

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 (REXEC) protocol and the Remote Shell (RSH) protocol. This chapter focuses on the remote execution functions that are available in the z/OS V1R8 Communications Server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, contain comprehensive descriptions of the individual parameters for setting up remote execution functions. They also include step-by-step checklists and supporting examples. It is not the intent of this book to duplicate the information in the manuals, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 9.1.2, "Additional information sources on remote execution and remote shell" on page 261.

This chapter discusses the following.

Section	Topic
9.1, "Overview" on page 258	Discusses the basic concepts of remote execution and remote shell
9.2, "Why remote execution is important" on page 261	Discusses key characteristics of remote execution and remote shell, and why it may be important in your environment
9.3, "The common design scenarios" on page 261	Presents commonly implemented remote execution and remote shell design scenarios, their dependencies, advantages, considerations, and our recommendations
9.4, "How remote execution is implemented" on page 267	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

9.1 Overview

As illustrated in Figure 9-1, REXEC and RSH are standard applications provided with the z/OS Communications Server to support remote execution services.

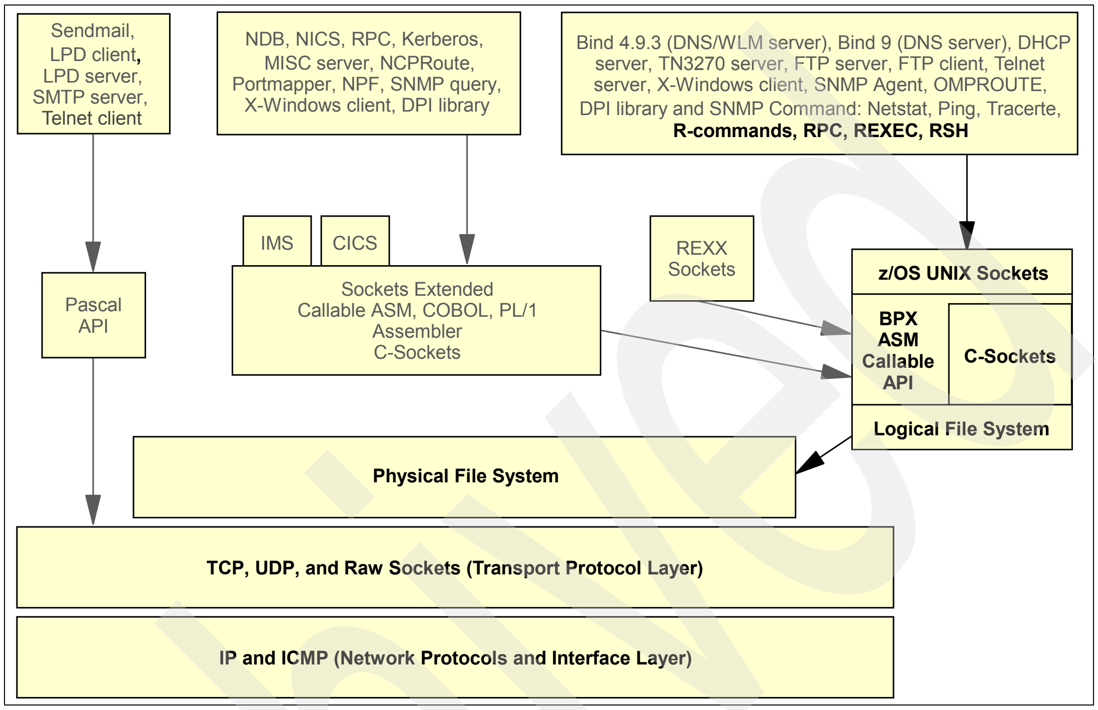

Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270 server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
REXX Sockets
z/OS UNIX Sockets
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

Figure 9-1 z/OS remote execution services

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)
- A z/OS UNIX remote shell server (orshd) and client (orsh)
- 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.

9.1.1 Basic concepts of remote execution and remote shell

Figure 9-2 on page 259 shows the relationship of the remote execution servers to the TCP/IP stack and the remote clients.

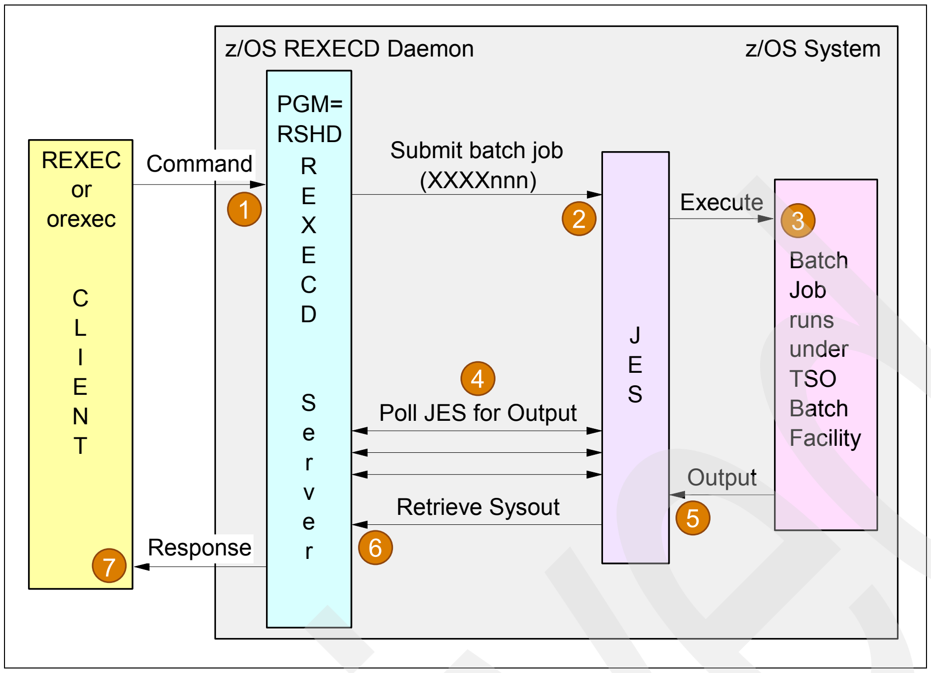


Figure 9-2 Remote execution client/server relationship

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, 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.

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 very 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.

Terminology

Terminology and acronyms can be very confusing. The documents referenced in 9.1.2, “Additional information sources on remote execution and remote shell” on page 261, 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 the HFS at /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 the HFS at /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 the 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 Additional information sources on remote execution and remote shell

Detailed information for the remote execution servers and protocols can be found in the following documents:

- *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.

9.2 Why remote execution is important

A client system may 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 may 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 may not have the software necessary to communicate with another system. An operator may not have a terminal or terminal emulation software that is compatible with the other platform in order 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.

9.3 The common design scenarios

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 this 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.

Their roles and recommendations are discussed in the following topics:

- TSO remote execution environment
- z/OS UNIX remote execution
- REXEC TSO client command using user ID/password
- REXEC TSO client command using the NETRC data set
- REXEC UNIX client command

9.3.1 TSO remote execution environment

This section provides an overview of the TSO remote execution environment. The following topics are discussed:

- Description of TSO remote execution
- TSO remote execution dependencies
- Advantages of TSO remote execution
- Considerations for using TSO remote execution

Description of TSO remote execution

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 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.

Advantages of TSO remote execution

A client system may 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.

Considerations for using TSO remote execution

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 modified. However, for an alternate approach to this see "Concurrent execution for TSO and z/OS UNIX remote execution servers" on page 273 in z/OS UNIX remote execution.

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 the *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for the details of using the security server to define these authorizations.

9.3.2 z/OS UNIX remote execution

This section provides an overview of the UNIX remote execution server (orexecd/rexecd) environment. The following topics are discussed:

- Description of the z/OS UNIX remote execution
- z/OS UNIX remote execution dependencies
- Advantages of using z/OS UNIX remote execution
- Considerations for using z/OS UNIX remote execution

Description of the z/OS UNIX remote execution

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.

Advantages of using z/OS UNIX remote execution

A client system may 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.

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.

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. In order 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 on running both servers concurrently, see "Implementation tasks for the UNIX OREXECD server" on page 271.

The z/OS UNIX remote execution clients cannot use a NETRC data set, and jobs cannot be submitted in batch.

9.3.3 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
- Dependencies of REXEC TSO with user ID and password
- Advantages of REXEC TSO with user ID and password
- Considerations for using REXEC TSO with user ID and password

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.

Advantages of REXEC TSO with user ID and 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.

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.

9.3.4 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 RECEC TSO using NETRC
- Dependencies of REXEC TSO using NETRC
- Advantages of REXEC TSO using NETRC
- Considerations for using REXEC TSO with NETRC

Description of RECEC TSO 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 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.

Advantages of REXEC TSO using NETRC

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 may 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.

Considerations for using REXEC TSO with 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.

9.3.5 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
- Dependencies of the REXEC UNIX client command
- Advantages of the REXEC UNIX client command

- Considerations for using the REXEC UNIX client command

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.

Advantages of the 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.

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.

9.3.6 Remote execution recommendations

In this section we discuss remote execution recommendations.

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 may 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 refer to 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 may 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 may 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.

9.4 How remote execution is implemented

Based upon our recommendations, we show implementation details for the following scenarios:

- TSO remote execution
- z/OS UNIX remote execution
- Using the REXEC TSO client command with a user ID/password
- Using the REXEC TSO client command with the NETRC data set
- Using the UNIX REXEC 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 319.
- Sample files for each application are listed in Appendix B, “Sample files provided with TCP/IP” on page 331.

9.4.1 TSO remote execution

The TSO remote execution server (RXSERVE) enables a remote client host to execute TSO commands on the local z/OS system. The following topics discuss the TSO REXECD server implementation:

- Implementation tasks for the TSO REXECD server
- Configuration examples for the TSO REXECD server
- Verification of the TSO REXECD server
- Problem determination for the TSO REXECD server

Implementation tasks for the TSO REXECD server

Perform the following steps to implement the RXSERVE started task:

1. Update the AUTOLOG and PORT statements in the PROFILE.TCPIP data set.
2. Determine which remote client commands to support: REXEC or RSH.
3. Optionally permit this RXSERVE task to be a surrogate job submitter by implementing security definitions using the IBM RACF security server or equivalent System Authorization Facility (SAF) interface system. (We do not show these steps).
4. Update the RXSERVE cataloged procedure in a system PROCLIB.
5. Create an optional user exit routine if the default //JOB and //EXEC JCL statements submitted by RXSERVE do not meet your installation’s JCL standards or requirements.
6. Create one or more TSO batch procedures to be submitted by RXSERVE.
7. Establish security access to JESSPOOL files, if necessary.

Configuration examples for the TSO REXECD server

Details of the implementation tasks are explained with configuration examples.

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
   RXSERVEB
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 RXSERVEB;for REXEC protocol
   514 TCP RXSERVEB for RSH protocol
```

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 RXSERVE

```
REXEC -l userid -p password -s 512 zos17b.ibm.com listalc
RSH -l userid/password -s 514 zos17b.ibm.com listalc
RSH -l userid -s 514 zos17b.ibm.com listalc
```

The RSH command without a password failed on our implementation with the message in Example 9-4, because the server did not have RACF surrogate authority defined.

Example 9-4 RSH command with missing password, error message

```
EZA4386E 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: You should 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 RXSERVE procedure

```
BROWSE    SYS1.PROCLIB(RXSERVEB) - 01.01             Line 00000000 Col 001 080
//RXSERVEB PROC MODULE='RSHD',
//          EXIT=,
//          TSOPROC=REXECTST,
//          MSGCLASS=A,
//          TSCLASS=H,
//          MAXCONN=512,
//          PREFIX=CS06,
//          PURGE=N,
//          IPV6=N,
//          SECLABEL=N,
//          TRACE=(LOG,SEND)
//*
//RXSERVEB 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 RXSERVE 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.

TSCLASS=x
   Output class for the //SYSTSPRT DD of the submitted job.
   Must be different from MSGCLASS

PURGE=x (Y or N)
   Should the submitted job's output be immediately purged when job completes?

IPV6=x (Y or N)
   Support communications with IPv6 addressing?

SECLABEL=x (Y or N)
   Add a security label to the job card of the submitted job?
   (Multilevel security related (MLS)).

TRACE=options
   LOG | NOLOG
   SEND | NOSEND
   CLIENT=clientid | ALLCLIENTS
   RESET

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

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).

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 JES interface to select the first file of the first output group from the job. For example, if the batch procedure specified is TSOPROC=REXECTST, Example 9-7 shows the correct placement of the //SYSTSPRT DD statement within the JCL.

Example 9-7 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(DATAB30),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. Refer to Example 9-3 on page 268 to see 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

Verification of the TSO REXECD server

RXSERVE can be started by an automations package, by a manual command, 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 9-8.

Example 9-8 RXSERVE startup messages

```
$$HASP100 RXSERVEB ON STCINRDR
IEF695I START RXSERVEB WITH JOBNAME RXSERVEB IS ASSIGNED TO USER TCPIP GROUP TCPGRP
$HASP373 RXSERVEB STARTED
IEF403I RXSERVEB - STARTED - TIME=10.41.11
EZA4400I Trace options: NOLOG,NOSEND,ALLCLIENTS
EZA4404I Parameters: MSGCLASS=H,TSCLASS=A,TSOPROC=REXECTST,MAXCONN=512,EXIT=REXECEXT
```

Problem determination for the TSO REXECD server

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 9-9.

Example 9-9 Modifying RXSERVE parameters dynamically

```
F RXSERVEB,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
```

9.4.2 z/OS UNIX remote execution

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 following topics discuss the UNIX OREXECD server implementation:

- Implementation tasks for the UNIX OREXECD server
- Configuration examples for the UNIX OREXECD server
- Verification of the UNIX OREXECD server
- Concurrent execution for TSO and z/OS UNIX remote execution servers

Implementation tasks for the UNIX OREXECD server

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.

Configuration examples for the UNIX OREXECD server

The orexecd record in the INETD /etc/inetd.conf file could look like one of the entries in Example 9-10, depending on the options that may 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 9-10 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
```

An actual record for **rexecd** in an /etc/inetd.conf file is shown in Example 9-11.

Example 9-11 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 BPXOINIT /usr/sbin/otelnetd otelnetd -l -m -t -c 10800
#shell     stream tcp nowait BPXOINIT /usr/sbin/rshd rshd -LV
#login     stream tcp nowait BPXOINIT /usr/sbin/rlogind rlogind -m
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd -LV
#finger    stream tcp nowait BPXOINIT /usr/sbin/fingerd fingerd
#smtp     stream tcp nowait BPXOINIT /usr/lib/sendmail sendmail -bn
#talk      dgram  udp wait   BPXOINIT /usr/sbin/talkd talkd
```

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. Notice the default port for shell/rsh/orsh is 514, as shown in Example 9-12.

Example 9-12 /etc/services entries for exec and shell

```
#
# 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
```

Verification of the UNIX OREXECD 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, as shown in Example 9-13.

Example 9-13 NETSTAT display shows INETD listening on port 512

```
MVS TCP/IP NETSTAT CS V1R8 TCPIP Name: TCPIPB         00:24:38
User Id  Conn     Local Socket           Foreign Socket         State
-------  ----     ------------           --------------         -----
BPXOINIT 00000069 0.0.0.0..10007         0.0.0.0..0             Listen
IMWEBSRV 0000002C 0.0.0.0..80            0.0.0.0..0             Listen
INETD    00000018 0.0.0.0..512           0.0.0.0..0             Listen
INETD    00000017 0.0.0.0..23            0.0.0.0..0             Listen
SYSLOGD3 00000013 0.0.0.0..587           *..*                   UDP
SNMPDB   00000023 0.0.0.0..1028          0.0.0.0..0             LISTEN
SNMPDB   00000022 0.0.0.0..161           *..*                   UDP
```

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 may 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. This allows them to listen on all defined IP addresses. However, this prevents both the TSO and z/OS UNIX remote execution servers from listening on the same port, and one of the servers would have 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. 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.

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 9-14.

Example 9-14 VIPA for BIND of `orexecd` *and orsh*

```
DEVICE STAVIPA1    VIRTUAL 0
 LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
;
   HOME
      10.20.4.234   IUTIQDF4LNK
      10.20.4.235   IUTIQDF5LNK
      10.20.5.236   IUTIQDF6LNK
      10.20.1.230   STAVIPA1LNK
      10.20.2.232    OSA2080LNK
      10.20.3.233    OSA20A0LNK
      10.20.2.234    OSA20C0LNK
      10.20.3.235    OSA20E0LNK
   ;
OR
   VIPADYNAMIC
     ;-----------------------------------------------------------------
     ;  Set aside 14 addresses for use with BIND and SIOCSVIPA IOCTL    -
     ;              (10.20.10.241 thru .254)                           -
     ;-----------------------------------------------------------------
      VIPARANGE  DEFINE  255.255.255.240  10.20.10.240
   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 9-15.

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 10.20.10.242 to the z/OS UNIX remote execution servers. Requests to ports 512 or 514 with a destination IP address that is not 10.20.10.242 are directed to the TSO remote execution server.

Example 9-15 BIND-specific assignments for orexecd and orshd

```
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    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.20.10.241     ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server
    25 TCP SMTPB                         ; SMTP Server
    53 TCP NAMED9B                       ; Domain Name Server
    53 UDP NAMED9B                       ; Domain Name Server
   111 TCP PORTMAPB                      ; Portmap Server
   111 UDP PORTMAPB                      ; 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 10.20.10.242        ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.20.10.242        ; 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
   750 TCP MVSKERBB                      ; Kerberos
   750 UDP MVSKERBB                      ; Kerberos
   751 TCP ADM@SRVB                      ; Kerberos Admin Server
   751 UDP ADM@SRVB                      ; Kerberos Admin Server
```

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 and both RSH servers are listening on port 514. INETD is now listening for the z/OS UNIX remote execution servers, in this case, on the VIPA address only. This is shown in Example 9-16.

Example 9-16 NETSTAT verifies BIND-specific listeners

```
MVS TCP/IP NETSTAT CS V1R8 TCPIP NAME: TCPIPB 00:50:41
User Id  Conn       Local Socket           Foreign Socket        State
-------  -------- ----------------------  ---------------------- --------
INETD2   0000000D 10.20.10.242..514       0.0.0.0..0            Listen
INETD2   0000000E 10.20.10.242..512       0.0.0.0..0            Listen
```

```
RXSERVEB 00000019 0.0.0.0..514            0.0.0.0..0            Listen
RXSERVEB 00000018 0.0.0.0..512            0.0.0.0..0            Listen
```

9.4.3 Using the REXEC TSO client command with a user ID/password

This section shows how to use the REXEC TSO client command with a user ID and password specified. The following topics discuss specifying the user ID and password on the REXEC command:

- Implementation tasks for including user ID and password
- Configuration examples for including a user ID and password
- Verification of using user ID and password
- Problem determination for using a user ID and password

Implementation tasks for including user ID and password

Use the REXEC command to execute a command on a remote host and receive the results on the local system, as shown in Example 9-17.

Example 9-17 REXEC TSO client command format

```
REXEC ? -d -n -l userid -p password -s portnum -t translate_file remote_host command
```

The *remote_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 -d, -l, -p, and -s are case sensitive and must be entered in lowercase letters. The user ID and password values may be case sensitive, depending on the remote operating system requirements.

Use a question mark (?) to obtain command help, as shown in Example 9-18.

Example 9-18 REXEC TSO client command help

```
rexec: no command given.
Usage: rexec  -? -d -l <usr> -p <pwd> -n -s <port> -t <fn> Fhost cmd
 options: -
         -?          display this message.
         -d          turn on debug tracing.
         -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
```

Use -d to activate debug tracing.

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 9-19.

Example 9-19 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
```

Configuration examples for including a user ID and password

REXEC TSO requests can be submitted from the TSO command line or from a batch job. These methods include:

- Submitting REXEC TSO requests from TSO including a user ID and password
- Submitting REXEC TSO requests in batch including a user ID and password

Submitting REXEC TSO requests from TSO 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 9-20.

Example 9-20 REXEC TSO client command with a user ID and password

```
READY
rexec -l CS06 -p poi1uytr 10.20.1.230 netstat home
```

Submitting REXEC TSO requests in batch including 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 9-17 on page 275.

The command can be entered as a parameter on the EXEC JCL statement, as shown in Example 9-21.

Example 9-21 Batch REXEC TSO client command via EXEC PARM=

```
BROWSE    CS06.TCPPARMS(JOBRX1) - 01.04            Line 00000000 Col 001 080
//CS06Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS06
//STEP1    EXEC PGM=REXEC,REGION=0M,
//        PARM='-l CS06 -p poi1uytr 10.20.1.230 netstat home'
//SYSPRINT DD DSN=CS06.NETSTAT.HOME,DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
```

Verification of using 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 you get back on your TSO session, as shown in Example 9-22.

Example 9-22 Results of REXEC TSO client command from a TSO session

```
MVS TCP/IP NETSTAT CS V1R8 TCPIP Name: TCPIPB         21:41:02
Home address list:
LinkName:   IUTIQDF4LNK
  Address:  10.20.4.234
    Flags:
LinkName:   IUTIQDF5LNK
  Address:  10.20.4.235
    Flags:
LinkName:   IUTIQDF6LNK
  Address:  10.20.5.236
    Flags:
LinkName:   STAVIPA1LNK
  Address:  10.20.1.230
    Flags:  Primary
LinkName:   OSA2080LNK
  Address:  10.20.2.232
    Flags:
LinkName:   OSA20A0LNK
  Address:  10.20.3.233
    Flags:
***
LinkName:   OSA20C0LNK
  Address:  10.20.2.234
    Flags:
LinkName:   OSA20E0LNK
  Address:  10.20.3.235
    Flags:
LinkName:   VIPL0A140A15
  Address:  10.20.10.21
    Flags:
LinkName:   VIPL0A140A16
  Address:  10.20.10.22
    Flags:
LinkName:   VIPL0A140A17
  Address:  10.20.10.23
    Flags:
LinkName:   LOOPBACK
  Address:  127.0.0.1
    Flags:
IntfName:   LOOPBACK6
  Address:  ::1
    Type:   Loopback
    Flags:
***
```

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 9-23.

Example 9-23 Results of REXEC TSO client command from a batch job

```
BROWSE    CS06.NETSTAT.HOME                          Line 00000000 Col 001 080
READY
netstat home
MVS TCP/IP NETSTAT CS V1R8 TCPIP Name: TCPIPB          02:46:24
Home address list:
LinkName:   IUTIQDF4LNK
  Address:  10.20.4.234
    Flags:
LinkName:   IUTIQDF5LNK
  Address:  10.20.4.235
    Flags:
LinkName:   IUTIQDF6LNK
  Address:  10.20.5.236
    Flags:
LinkName:   STAVIPA1LNK
  Address:  10.20.1.230
    Flags:  Primary
LinkName:   OSA2080LNK
  Address:  10.20.2.232
    Flags:
LinkName:   OSA20A0LNK
  Address:  10.20.3.233
    Flags:
LinkName:   OSA20C0LNK
  Address:  10.20.2.234
    Flags:
LinkName:   OSA20E0LNK
  Address:  10.20.3.235
    Flags:
LinkName:   VIPL0A140A15
  Address:  10.20.10.21
    Flags:
LinkName:   VIPL0A140A16
  Address:  10.20.10.22
    Flags:
LinkName:   VIPL0A140A17
  Address:  10.20.10.23
    Flags:
LinkName:   LOOPBACK
  Address:  127.0.0.1
    Flags:
IntfName:   LOOPBACK6
  Address:  ::1
    Type:   Loopback
    Flags:
READY
END
```

Problem determination for using a 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 may not process the command stream properly.

Submitting a long-running command may cause the REXEC program to end abnormally with a 522 system abend code. This can be avoided 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 9-24.

Example 9-24 Executing a remote command having an imbedded slash (/): PARM=

```
//CS06C    JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS06
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='/-l oper6 -p cold2day 10.20.10.242 ls ./lpp/samples/*'
//SYSPRINT DD   DSN=CS06.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 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 may not allow the REXEC client program access to the stack.
  - REXEC may not be configured, or may 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 may not be correctly set up for your user ID to log on to the remote server.
  - The security may not be correctly set up for your user ID to execute the requested command on the server.

9.4.4 Using the REXEC TSO client command with the NETRC data set

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. The following topics discuss omitting the user ID and password from the REXEC command:

- Implementation tasks for omitting password and using NETRC
- Configuration examples for omitting password and using NETRC
- Verification for omitting password and using NETRC
- Problem determination for omitting password and using NETRC

Implementation tasks for omitting password and using NETRC

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 ? -d -n -s portnum -t translate_file remote_host command
```

A full description of these parameters is given in "Implementation tasks for including user ID and password" on page 275.

Configuration examples for omitting password and using NETRC

REXEC TSO requests can be submitted from the TSO command line or from a batch job. These methods include:

- Submitting REXEC TSO requests from TSO without a user ID and password
- Submitting REXEC TSO requests in batch without a user ID and password
- Preparing and using the NETRC data set

Submitting REXEC TSO requests from TSO 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 9-22 on page 277.

Example 9-26 REXEC TSO client command without a user ID and password

```
READY
rexec 10.20.1.230 netstat conn
```

Submitting REXEC TSO requests in batch without 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 described in "Implementation tasks for including user ID and password" on page 275.

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    CS06.TCPPARMS(JOBRX3) - 01.03             Line 00000000 Col 001 080
//CS06Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS06
```

```
//STEP1    EXEC PGM=REXEC,REGION=0M,
//     PARM='10.20.1.230 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 as well as 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. The NETRC data set used in our scenarios is shown in Example 9-29.

Example 9-29 Sample NETRC data set

```
machine TCPSYTO.ibm.com login TSTEM001 password zse43wa
machine 10.20.1.230 login CS06 password POI1UYTR
machine 10.20.1.241 login CS06 password POI1UYTR
machine 10.20.1.221 login CS06 password POI1UYTR
machine LPAR2 login CS06 password mko09ijn
machine system3.ibm.com login mngr27 password my03pswd
machine system5.ibm.com login mngr27 password my05pswd
machine system7.ibm.com login mngr27 password my07pswd
```

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    CS06.TCPPARMS(JOBRX3) - 01.03              Line 00000000 Col 001 080
//CS06Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS06
//STEP1    EXEC PGM=REXEC,REGION=0M,
//     PARM='10.20.1.230 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

```
//CS06F    JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS06
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='10.20.1.230 netstat conn'
//SYSPRINT DD   DSN=CS06.NETSTAT.CONN,DISP=SHR
//NETRC    DD   DSN=CS06.OVERRIDE.NETRC,DISP=SHR
```

Verification for omitting password and 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

```
MVS TCP/IP NETSTAT CS V1R8 TCPIP Name: TCPIPB          21:57:39
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 CS06Z    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 ---- MONDAY,    21 AUG 2006 ----
23.22.51 JOB07179  IRR010I  USERID CS06     IS ASSIGNED TO THIS JOB.
23.23.25 JOB07179  ICH70001I CS06     LAST ACCESS AT 23:19:07 ON MONDAY, AUGUST
23.23.25 JOB07179  $HASP373 CS06Z    STARTED - INIT 3    - CLASS C - SYS SC30
        1 //CS06Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS0
        2 //STEP1    EXEC PGM=REXEC,REGION=0M,
          //     PARM='-d 10.20.1.230 netstat home'
        3 //SYSPRINT DD SYSOUT=*
        4 //SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
ICH70001I CS06     LAST ACCESS AT 23:19:07 ON MONDAY, AUGUST 21, 2006
IEF236I ALLOC. FOR CS06Z STEP1
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I 803B ALLOCATED TO SYSTCPD
IEF237I 803B ALLOCATED TO SYS00004
Established affinity with TCPIPB
EZA4809I  Using NETRC file //'CS06.NETRC'.
EZA4762I  MACHINE : 10.20.1.230
EZA4763I  LOGIN   : CS06
EZA4764I  PASSWORD: ******
EZA4801I  MVS TCP/IP REXEC CS V1R8
EZA4775I  Calling function rexec_af with the following:
Host: 10.20.1.230   user: CS06   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 CS064    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 ---- MONDAY, 21 AUG 2006 ----
23.22.57 JOB07180  ICH70001I CS06     LAST ACCESS AT 23:22:57 ON MONDAY, AUGUST
23.22.57 JOB07180  $HASP373 CS064    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  -CS064    TSO      GENERAL     00    466    .00    .00    .00
23.22.57 JOB07180  -CS064    ENDED.  NAME-                     TOTAL CPU TIME=
23.22.57 JOB07180  $HASP395 CS064    ENDED
------ JES2 JOB STATISTICS ------
  21 AUG 2006 JOB EXECUTION DATE
           11 CARDS READ
          108 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
            4 SYSOUT SPOOL KBYTES
         0.00 MINUTES EXECUTION TIME
        1 //CS064    JOB CS06,
          // USER=CS06,
          // 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    CS06.NETSTAT.CONN                          Line 00000000 Col 001 080
MVS TCP/IP NETSTAT CS V1R8 TCPIP Name: TCPIPB          21:59:06
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: *..*
```

Problem determination for omitting password and 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 may not process the command stream properly.

Submitting a long-running command may cause the REXEC program to end abnormally with a 522 system abend code. This can be avoided 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 9-36.

Example 9-36 Executing a remote command having an imbedded slash (/)

```
//CS06E JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS06
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='/-l oper6 -p cold2day 10.20.10.242 ls ./lpp/samples/*'
//SYSPRINT DD   DSN=CS06.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 may not allow the REXEC client program access to the stack.
  - REXEC may not be configured, or may 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 may not be correctly set up for your user ID to log on to the remote server.
  - The security may not be correctly set up for your user ID to execute the requested command on the server.

9.4.5 Using the UNIX REXEC client command

This section shows how to use the REXEC UNIX client command (**rexec**/**orexec**). The following topics discuss using the rexec/orexec z/OS UNIX client command:

- Implementation tasks for the UNIX REXEC client command
- Configuration examples for the UNIX REXEC client command
- Verification for using the UNIX REXEC client command
- Problem determination for the UNIX REXEC client command

Implementation tasks for the UNIX REXEC 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 9-37. The **rexec** command is a synonym for the **orexec** command in the z/OS UNIX shell. They both have the same format.

Example 9-37 REXEC UNIX client command format

```
orexec ? -d -l userid -p password -s portnum remote_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 may be case sensitive, depending on the remote operating system requirements.

Use a question mark (?) to obtain command help, as shown in Example 9-38.

Example 9-38 REXEC UNIX client command help

```
CS06 @ SC30:/u/cs06>rexec ?
EZYRC23E  Command is missing
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
CS06 @ SC30:/u/cs06>
```

Use -d to activate debug tracing.

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.

Use *remote_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 in order to complete. The **rexec**/**orexec** command cannot interact with you after you enter data in the command format.

Configuration examples for the UNIX REXEC client command

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 9-39.

Example 9-39 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 9-40.

Example 9-40 REXEC UNIC client command with user ID and password

```
READY
orexec -l CS06 -p poi1uytr -s 512 10.20.10.242 netstat home
```

Verification for using the UNIX REXEC client command

You can verify the results of the REXEC UNIX client command by reviewing the response you get back, as shown in Example 9-41.

Example 9-41 Results of REXEC UNIX client command

```
CS06 @ SC30:/u/cs06>rexec -l cs06 -p poi1uytr -s 512 9.12.4.211 netstat home
MVS TCP/IP NETSTAT CS V1R8 TCPIP Name: TCPIP           21:27:13
Home address list:
LinkName:   OSA2040LNK
  Address:  9.12.4.211
    Flags:  Primary
LinkName:   STAVIPA1LNK
  Address:  9.12.4.212
    Flags:
LinkName:   LOOPBACK
  Address:  127.0.0.1
    Flags:
IntfName:   LOOPBACK6
  Address:  ::1
    Type:   Loopback
```

Problem determination for the UNIX REXEC 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 may not allow the REXEC client program access to the stack.
  - REXEC may not be configured, or may 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 may not be correctly set up for your user ID to log on to the remote server.
  - The security may not be correctly set up for your user ID to execute the requested command on the server.

10

Domain Name Services

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 V1R8 Communications Server. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771.

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains comprehensive descriptions of the individual parameters for setting up a DNS server. It also includes step-by-step checklists and supporting examples. It is not the intent of this book to duplicate the information in the referenced manual, but to complement it with practical implementation scenarios that can be useful in your environment. For complete details, we encourage you to review the documents referred to in 10.1.2, “Additional information sources for DNS” on page 292.

This chapter discusses the following.

Section	Topic
10.1, “Overview” on page 290	Discusses the basic concepts of Domain Name Services
10.2, “Why Domain Name Services is important” on page 292	Discusses key characteristics of a DNS and why it may be important in your environment
10.3, “The common design scenarios for DNS” on page 292	Presents commonly implemented DNS design scenarios, their dependencies, advantages, considerations, and our recommendations
10.4, “How a caching-only DNS server is implemented” on page 296	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

10.1 Overview

As illustrated in Figure 10-1, the DNS name server is one of the standard applications provided with the z/OS Communications Server.

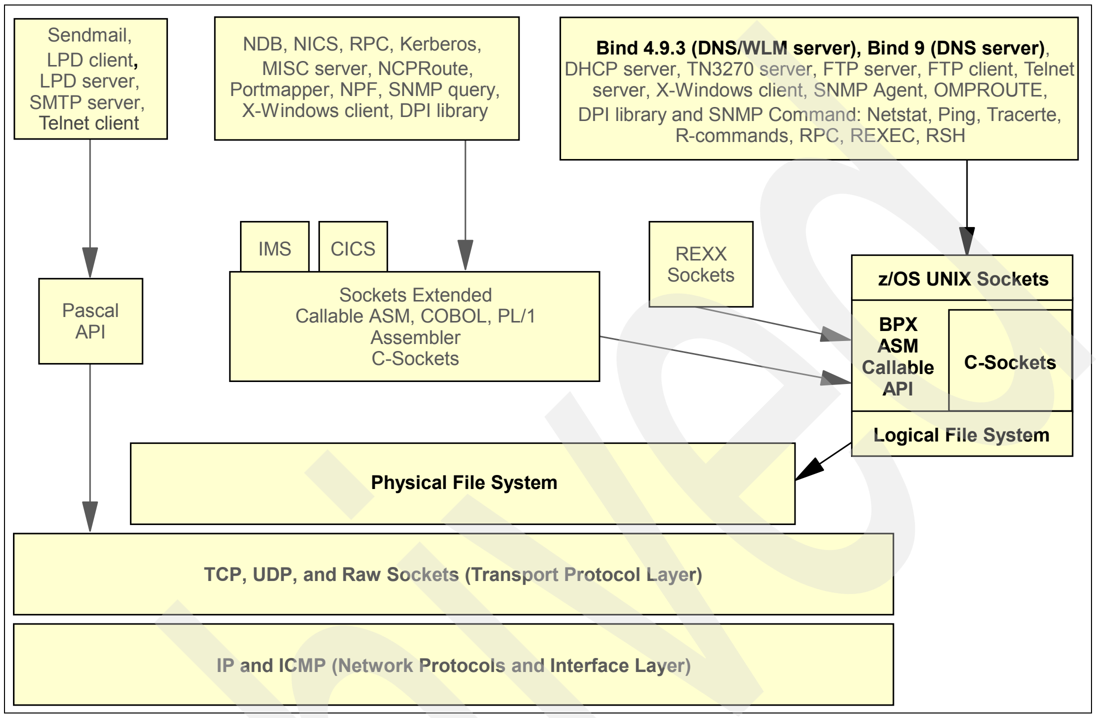

Sendmail, LPD client, LPD server, SMTP server, Telnet client
NDB, NICS, RPC, Kerberos, MISC server, NCPRoute, Portmapper, NPF, SNMP query, X-Windows client, DPI library
Bind 4.9.3 (DNS/WLM server), Bind 9 (DNS server), DHCP server, TN3270 server, FTP server, FTP client, Telnet server, X-Windows client, SNMP Agent, OMPROUTE, DPI library and SNMP Command: Netstat, Ping, Tracerte, R-commands, RPC, REXEC, RSH
IMS
CICS
REXX Sockets
z/OS UNIX Sockets
Pascal API
Sockets Extended Callable ASM, COBOL, PL/1 Assembler C-Sockets
BPX ASM Callable API
C-Sockets
Logical File System
Physical File System
TCP, UDP, and Raw Sockets (Transport Protocol Layer)
IP and ICMP (Network Protocols and Interface Layer)

Figure 10-1 DNS services

Name servers provide names and IP addresses to clients called *resolvers*. The information helps to cross-reference an IP address to a name, and the name to an IP address.

10.1.1 Basic concepts of Domain Name Services

The relationships between resolvers (the clients) and DNS servers are illustrated in Figure 10-2 on page 291.

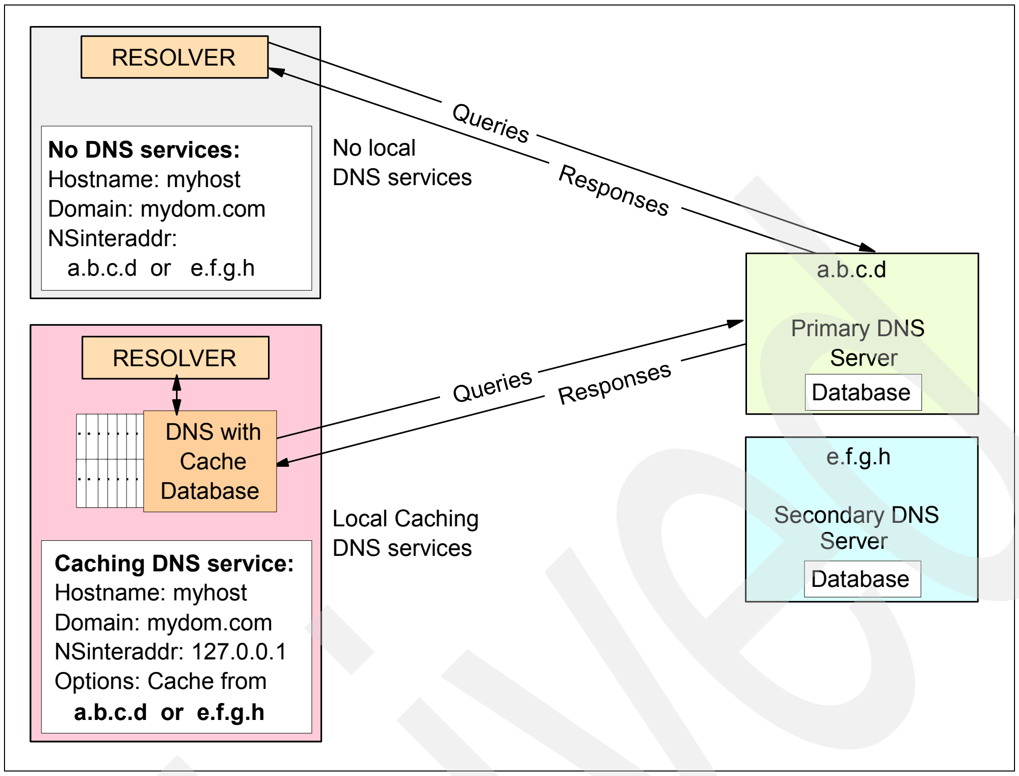


Figure 10-2 Caching DNS server and the resolver

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
itsodns1.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 *slave* (backup) for another part.

10.1.2 Additional information sources for DNS

Detailed information about the DNS server and protocol can be found in the following documents:

- *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
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341

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.

10.2 Why Domain Name Services is important

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.

10.3 The common design scenarios for DNS

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 two types of DNS servers:

- Authoritative DNS server
- Caching-only DNS server

10.3.1 Authoritative DNS server

An authoritative DNS server is responsible for one or more zones, managing the names and IP address associations within that zone. Here we discuss the following:

- Description of an authoritative DNS server
- Dependencies of an authoritative DNS server
- Advantages of an authoritative DNS server
- Considerations for using an authoritative DNS server

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.

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 slave 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 slave (secondary). Each zone must have only one master name server, and it should have at least one slave 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 (due to 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 section 2.5.5.10 in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Advantages of an authoritative DNS server

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.

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 above, you need to design your DNS service for sufficient scalability and availability to meet your business requirements. Unavailability of DNS services 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 services could require significant processor resources, especially if you implementation requires security (DNSSEC, authenticating DNS data with digital signatures) implementation.

10.3.2 Caching-only DNS server

All name servers cache (store) the data they receive in response to a query. As the name suggests, however, 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.

In this section we discuss the following:

- Description of a caching-only DNS server
- Dependencies of a caching-only DNS server
- Advantages of a caching-only DNS server
- Considerations for using a caching-only DNS server

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.

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.

Advantages of a caching-only DNS server

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: You should 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. See Example 10-14 on page 303 for an example of this setup).

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: You should 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 may require the address you specify for the name server to match the source IP address in the response from the name server. For example, if a static VIPA address is specified as the address of the name server, and IPCONFIG SOURCEVIPA is not specified in PROFILE.TCPIP, then nslookup on some platforms will discard the returned packet because it will have the destination address of the physical interface instead of the VIPA interface.

10.3.3 Recommendations for implementing Domain Name Services

Most organizations have little need to design, implement, manage, and administer their own name space and authoritative DNS services for their IBM eServer zSeries 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 may play an important role, especially to improve the performance of z/OS Web servers and WebSphere® applications. Caching DNS information on board can reduce CPU utilization and increase responsiveness to the numerous Web transactions that require DNS services.

We therefore focus our implementation on setting up and using a BIND 9 Caching-only DNS on the z/OS system.

Because we recommend a BIND 9 DNS, we need to discuss some BIND history and point out some compatibility issues between BIND 4.9.3 and BIND 9:

- z/OS V1R2 Communications Server BIND 9.1 was the first implementation of BIND 9 on the z/OS platform. It was a complete rewrite of the name server and associated utilities. This allowed IPv6-type records in zone data. It also introduced better transaction security among servers and clients, as well as zone data authentication capability. It introduced the rndc utility to replace and complement UNIX signals for name server local and remote control.
- The configuration file for the name server changed in both name and syntax between BIND 4.9.3 and BIND 9. The *dnsmigrate* tool aids in converting a BIND 4.9.3 named.boot file into a BIND 9 named.conf file. This utility is available as of z/OS V1R2.
- The BIND 9.2 name server was introduced in z/OS V1R4. It makes DNS server-to-server and client-to-server IPv6 connections possible, adding new name server configuration options for IPv6 connections and tuning. BIND 9.2 also provides a new 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. The z/OS DHCP server is compatible with the z/OS BIND 4.9.3 name server, but is incompatible with the BIND 9 name server.

Note: There are a number of considerations to be aware of when running the BIND 9 server in a multiple stack environment (CINET). Refer to the BIND 9 advanced topics section in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details on these restrictions.

10.4 How a caching-only DNS server is implemented

Based upon the recommendations in 10.3.3, “Recommendations for implementing Domain Name Services” on page 295, we show implementation details for the BIND 9 caching-only DNS server.

10.4.1 BIND 9 caching-only DNS server setup

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. The steps for configuring a caching-only server are discussed in the following sections:

- Implementation tasks for a caching-only DNS server
- Configuration examples for a caching-only DNS server
- Verification for a caching-only DNS server
- Problem determination for a caching-only DNS server

Implementation tasks for a caching-only DNS server

The *z/OS Communications Server: IP Configuration Guide*, SC31-8775, contains a comprehensive discussion on setting up the BIND 9 Caching-only server. Refer to that manual for details. A summarized checklist is provided here to assist with understanding the above reference. Most of the configuration files that resulted from performing these steps on our test system are listed in “Configuration examples for a caching-only DNS server” on page 298.

The following steps are necessary to set up and activate a caching-only server:

1. Create a BIND 9 configuration file for the server.
2. Create an environment variables file for the server.
3. Update the TCP/IP stack profile port reservation list for the server.
4. Create the name server start procedure.
5. Create the hints file (root server).
6. Create the loopback file.
7. Configure logging options for the server.
8. Implement the syslog daemon (syslogd) or alternative log file.
9. Determine and set the run mode of the server: swappable or nonswappable.
10. Start the name server task.

Create a BIND 9 configuration file for the server

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/dnsdatb30, we copied the sample file to /etc/dsndatb30/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 NAMEDDD, and the forked job resulted in a name of NAMEDDD1. 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 resolver config file provides the NSINTERADDR/NAMESERVER IP address for the BIND 9 server so it can make that address its primary loopback address. The NSINTERADDR address should be the 127.0.0.1 address or another address in your HOME list that is defined to be a loopback address. The STDENV file that we used is shown in Example 10-1.

Example 10-1 Sample STDENV file

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIP
_BPX_JOBNAME=NAMEDDD
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DAT30TST)'
DNS_VERSION=v9
```

Update the TCP/IP stack profile port reservation list for the server

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 to avoid any name conflicts in the future, especially if you change the name of the started task.

```
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.

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 at `ftp://ftp.rs.internic.net` and retrieve the file named.root from the domain subdirectory.

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 null.

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/dnsdatb30, we copied the sample file to /etc/dnsdatb30.db.loopback.v9 and customized it to reflect our domain name. For a comprehensive discussion on the loopback file for caching servers, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Configure logging options for the server

A wide variety of logging options for the name server can be configured via 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, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for a discussion about how to use logging. For details on the syntax and format of the logging statement, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

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 details in setting syslogd, refer to the syslogd chapter in this book.

Determine and set 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.

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 via automations. We started our NAMED9B procedure using the MVS START command:

```
S NAMED9B
```

Configuration examples for a caching-only DNS server

We created a working directory (dnsdatb30) in the HFS 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 dynamically created by the server task. We coded a directory option statement pointing to the desired directory. A directory list of /etc/dnsdatb30 resulted in the display shown in Example 10-2.

Example 10-2 Listing of the HFS directory for the NAMED9B server, /etc/dnsdatb30

```
Directory List
EUID=3606   /SC30/etc/dnsdatb30/
  Type  Perm  Permission  Changed-EST5EDT   Owner      Filename    Row 1 of 8
_ Dir    777   rwxrwxrwx  2006-08-24 20:36  CS06       .
_ Dir    777   rwxrwxrwx  2006-08-24 17:57  HAIMO      ..
_ File   777   rwxrwxrwx  2006-08-24 02:10  CS06       db.loopback.v9
_ File   666   rw-rw-rw-  2006-08-24 20:33  BPXROOT    log.log
b File   777   rwxrwxrwx  2006-08-24 17:47  CS06       named9b.conf
b File   777   rwxrwxrwx  2006-08-24 18:14  CS06       named9b.env
_ File   644   rw-r--r--  2006-08-24 20:36  BPXROOT    named9b.pid
_ File   777   rwxrwxrwx  2006-08-24 03:27  CS06       named9b.stats
```

Example 10-3 is the configuration file for a caching-only server.

Example 10-3 Configuration file for the NAMED9B task

```
BROWSE -- /SC30/etc/dnsdatb30/named9b.conf --------- Line 00000000 Col 001 050
#  named.conf file for named9b for sc30
```

```
options {
   directory "/etc/dnsdatb30";
   pid-file "/etc/dnsdatb30/named9b.pid";
   statistics-file "/etc/dnsdatb30/named9b.stats";
   forward only ;
   forwarders {9.12.6.7; 9.12.12.7; 9.12.10.7; };
   max-ncache-ttl 1800;
   max-cache-ttl 36000;
};

logging {
   channel "myfile" {
      file "/etc/dnsdatb30/log.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 "db.loopback.v9";
};
zone "." in {
   type hint;
   file "/etc/dnsdatb30/db.cache";
};
```

Example 10-4 is the environment variable file.

Example 10-4 Environment variables file for BIND 9 DNS server

```
BROWSE -- /SC30/etc/dnsdatb30/named9b.env ---------- Line 00000000 Col 001 045
_BPXK_SETIBMOPT_TRANSPORT=TCPIP
_BPX_JOBNAME=NAMED9B
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DAT30TST)'
DNS_VERSION=v9
```

Example 10-5 is the loopback file.

Example 10-5 Loopback file for BIND 9 DNS server

```
BROWSE -- /SC30/etc/dnsdatb30/db.loopback.v9 ------- Line 00000000 Col 001 065
; Default TTL value
$TTL 86400
0.0.127.in-addr.arpa. IN SOA  itsodns.itso.ibm.com              (
     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.
```

The PORT reservation list would look like Example 10-6.

Example 10-6 Port reservation for the DNS server

```
PORT
   53 TCP OMVS
   53 UDP OMVS
```

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 10-7.

Example 10-7 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.X
//            /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=*
//CEEDUMP  DD SYSOUT=*
```

Sample messages issued at startup are shown in Example 10-8.

Example 10-8 NAMED9 startup messages

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 0  --  N O D E
20.36.49 STC08337 ---- MONDAY,    24 AUG 2006 ----
20.36.49 STC08337  IEF695I START NAMED9B  WITH JOBNAME NAMED9B  IS ASSIGNED TO U
20.36.49 STC08337  $HASP373 NAMED9B  STARTED
20.36.50 STC08337  $HASP395 NAMED9B  ENDED
```

After startup, the process ID of the NAMED9B task is placed into the PID file, as shown in Example 10-9.

Example 10-9 PID file containing the process ID of the NAMED9B task

```
BROWSE -- /SC30/etc/dnsdatb30/named9b.pid ---------- Line 00000000 Col 001 005
65692
```

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 10-10.

Example 10-10 Stopping the NAMED9B task using the UNIX shell

```
kill -TERM 65692
or
kill -TERM $(cat /etc/dnsdatb30/named9b.pid)
```

```
or
P NAMED9B1
```

Verification for a caching-only DNS server

The startup of the server and its responsiveness can be verified as follows:

1. Verify that the name server started correctly.
2. Use netstat to verify that the server is listening on the intended port.
3. Verify that the name server can accept queries.
4. Browse the log to see the latest query activity just created.
5. Stop the caching-only DNS server to verify it will shut down cleanly.

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 10-11, 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 (refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781). For our test environment, we did not implement **rndc**.

Example 10-11 NAMED9 log immediately after startup

```
BROWSE -- /SC30/etc/dnsdatb30/log.log -------------- Line 00000000 Col 035 114
1327bef0: EZZ9126I loading configuration from '/etc/dnsdatb30/named9b.conf'
ng: 1327bef0: EZZ9573I set maximum stack size to 0: invalid file
ng: 1327bef0: EZZ9573I set maximum data size to 0: invalid file
 1327bef0: EZZ9719I none:0: open: /etc/rndc.key: file not found
e: 1327bef0: EZZ9112I couldn't add command channel 127.0.0.1#953: file not found
 1327bef0: EZZ9719I none:0: open: /etc/rndc.key: file not found
e: 1327bef0: EZZ9112I couldn't add command channel ::1#953: file not found
```

Use netstat to verify that the server is listening on the intended port

Netstat 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 10-12.

Example 10-12 Verifying NAMED9 connections with NETSTAT

```
netstat conn
.
NAMED9B1 0000C642 Listen
  Local Socket:   9.12.4.212..53
  Foreign Socket: 0.0.0.0..0
NAMED9B1 0000C640 Listen
  Local Socket:   9.12.4.211..53
  Foreign Socket: 0.0.0.0..0
NAMED9B1 0000C644 Listen
  Local Socket:   127.0.0.1..53
  Foreign Socket: 0.0.0.0..0
.
.NAMED9B1 0000C645 UDP
  Local Socket:   0.0.0.0..14619
  Foreign Socket: *..*
NAMED9B1 0000C646 UDP
  Local Socket:   ::..14620 (IPV6_ONLY)
  Foreign Socket: *..*
NAMED9B1 0000C641 UDP
```

```
  Local Socket:   9.12.4.212..53
  Foreign Socket: *..*
NAMED9B1 0000C643 UDP
  Local Socket:   127.0.0.1..53
  Foreign Socket: *..*
NAMED9B1 0000C63F UDP
  Local Socket:   9.12.4.211..53
  Foreign Socket: *..*
```

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 'SYS1.TCPPARMS(*DATA30*)' 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 10-13.

Example 10-13 Uses external DNS server

```
BROWSE    SYS1.TCPPARMS(DATA30) - 01.05              Line 00000000 Col 001 080
TCPIPJOBNAME TCPIP
HOSTNAME WTSC30
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIP
MESSAGECASE MIXED
NSINTERADDR 127.0.0.1
NSINTERADDR  9.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

alloc ddn(systcpd) dsn('sys1.tcpparms(data30)') shr
nslookup

 EZB3170I Default Server:  itsodns.itso.ibm.com
 EZB3172I Address:  9.12.6.7
 EZB3042I >
wtsc30
 EZB3170I Server:  itsodns.itso.ibm.com
 EZB3172I Address:  9.12.6.7
 EZB3170I Name:    wtsc30.ITSO.IBM.COM
 EZB3172I Address:  9.12.4.211

 EZB3042I >
wtsc31
 EZB3170I Server:  itsodns.itso.ibm.com
 EZB3172I Address:  9.12.6.7
 EZB3170I Name:    wtsc31.ITSO.IBM.COM
 EZB3172I Address:  9.12.4.213
 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). An example of this method uses file 'TCPIPB.TCPPARMS(*DAT30TST*). 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 10-14.

Example 10-14 Local caching DNS specified, loopback address

```
BROWSE    TCPIPB.TCPPARMS(DAT30TST) - 01.02         Line 00000000 Col 001 080
TCPIPJOBNAME TCPIP
HOSTNAME WTSC30
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIP
MESSAGECASE MIXED
NSINTERADDR  127.0.0.1   <=== Was 9.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

alloc ddn(systcpd) dsn('tcpipb.tcpparms(dat30tst)') 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:  9.12.4.211

 EZB3042I >
wtsc31
 EZB3170I Server:  localhost
 EZB3172I Address:  127.0.0.1

 EZB3110I Non-authoritative answer:
 EZB3170I Name:    wtsc31.ITSO.IBM.COM
 EZB3172I Address:  9.12.4.213

 EZB3042I >
```

Compare the server names and addresses between the two methods. Notice the first method used server *itsodns.itso.ibm.com* at *9.12.6.7*, 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*. This is shown in Example 10-15.

Example 10-15 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: 0
EZB3332I ;; QUESTIONS:
;;      wtsc30.ITSO.IBM.COM, type = A, class = IN

EZB3359I ;; ANSWERS:
wtsc30.ITSO.IBM.COM.    A       9.12.4.211

dig wtsc31
```

```
EZB3316I ;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 3
EZB3328I ;; Ques: 1,Ans: 1,Auth: 0,Addit: 0
EZB3332I ;; QUESTIONS:
;;      wtsc31.ITSO.IBM.COM, type = A, class = IN

EZB3359I ;; ANSWERS:
wtsc31.ITSO.IBM.COM.    A       9.12.4.213
```

Browse the log to see the latest query activity just created

Checking the log once again, as in Example 10-16, shows the results of the query activity (nslookup and dig) that was just performed.

Example 10-16 NAMED9 Log after using nslookup and dig for WTSC30 and WTSC31

```
BROWSE -- /SC30/etc/dnsdatb30/log.log -------------- Line 00000000 Col 035 114
1327bef0: EZZ9126I loading configuration from '/etc/dnsdatb30/named9b.conf'
ng: 1327bef0: EZZ9573I set maximum stack size to 0: invalid file
ng: 1327bef0: EZZ9573I set maximum data size to 0: invalid file
 1327bef0: EZZ9719I none:0: open: /etc/rndc.key: file not found
e: 1327bef0: EZZ9112I couldn't add command channel 127.0.0.1#953: file not found
 1327bef0: EZZ9719I none:0: open: /etc/rndc.key: file not found
e: 1327bef0: EZZ9112I couldn't add command channel ::1#953: file not found

 1327bef0: EZZ8828I client 127.0.0.1#14609: query: 1.0.0.127.in-addr.arpa IN PTR
 1327bef0: EZZ8828I client 127.0.0.1#14611: query: wtsc30.ITSO.IBM.COM IN A
 1327ec20: EZZ8828I client 127.0.0.1#14612: query: wtsc31.ITSO.IBM.COM IN A

1327ce00: EZZ8828I client 127.0.0.1#14650: query: . IN A
1327fb30: EZZ8828I client 127.0.0.1#14655: query: wtsc30.ITSO.IBM.COM IN A
1327fb30: EZZ8828I client 127.0.0.1#14657: 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 10-17.

Example 10-17 NAMED9 log immediately after shutdown of NAMED9 task

```
BROWSE -- /SC30/etc/dnsdatb30/log.log -------------- Line 00000000 Col 035 114
1327bef0: EZZ9126I loading configuration from '/etc/dnsdatb30/named9b.conf'
ng: 1327bef0: EZZ9573I set maximum stack size to 0: invalid file
ng: 1327bef0: EZZ9573I set maximum data size to 0: invalid file
 1327bef0: EZZ9719I none:0: open: /etc/rndc.key: file not found
e: 1327bef0: EZZ9112I couldn't add command channel 127.0.0.1#953: file not found
 1327bef0: EZZ9719I none:0: open: /etc/rndc.key: file not found
e: 1327bef0: EZZ9112I couldn't add command channel ::1#953: file not found

 1327bef0: EZZ8828I client 127.0.0.1#14609: query: 1.0.0.127.in-addr.arpa IN PTR
 1327bef0: EZZ8828I client 127.0.0.1#14611: query: wtsc30.ITSO.IBM.COM IN A
 1327ec20: EZZ8828I client 127.0.0.1#14612: query: wtsc31.ITSO.IBM.COM IN A

1327ce00: EZZ8828I client 127.0.0.1#14650: query: . IN A
1327fb30: EZZ8828I client 127.0.0.1#14655: query: wtsc30.ITSO.IBM.COM IN A
1327fb30: EZZ8828I client 127.0.0.1#14657: query: wtsc31.ITSO.IBM.COM IN A

 1327ec20: EZZ9131I shutting down
 1327ec20: EZZ9045I no longer listening on 9.12.4.211#53
 1327ec20: EZZ9045I no longer listening on 9.12.4.212#53
```

```
1327ec20: EZZ9045I no longer listening on 127.0.0.1#53
```

Problem determination for a caching-only DNS server

DNS server problems can be managed by using the following techniques:

- 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 rndc command from the OMVS shell 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 10-18.

Example 10-18 NAMED9 JCL: example of invalid PARM string and resulting message at startup

```
BROWSE    SYS1.PROCLIB(NAMED9X) - 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 0  --  N O D E
20.34.19 STC08336 ---- MONDAY,    24 AUG 2006 ----
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
```

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 2, “SYSLOGD” on page 97.

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 may 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 10-19.

Example 10-19 Starting NAMED9 from the z/OS UNIX shell to get early logging information

```
su

export _CEE_ENVFILE="/etc/ndsdatb30/named9b.env"

named -V v9 -g -c /etc/dnsdatb30/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 rndc command from the OMVS shell 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 on using the **rndc** command, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

10.5 Automated domain name registration (ADNR)

This application allows you to dynamically add/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 the entire sysplex, as well as names that represent individual systems within the sysplex.

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 is especially important because DNS/WLM and BIND 4.9.3 support will be removed in a future release. Also, it removes the restriction of 15 IPv4 addresses and adds support for IPv6 addresses.

Basically, 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.

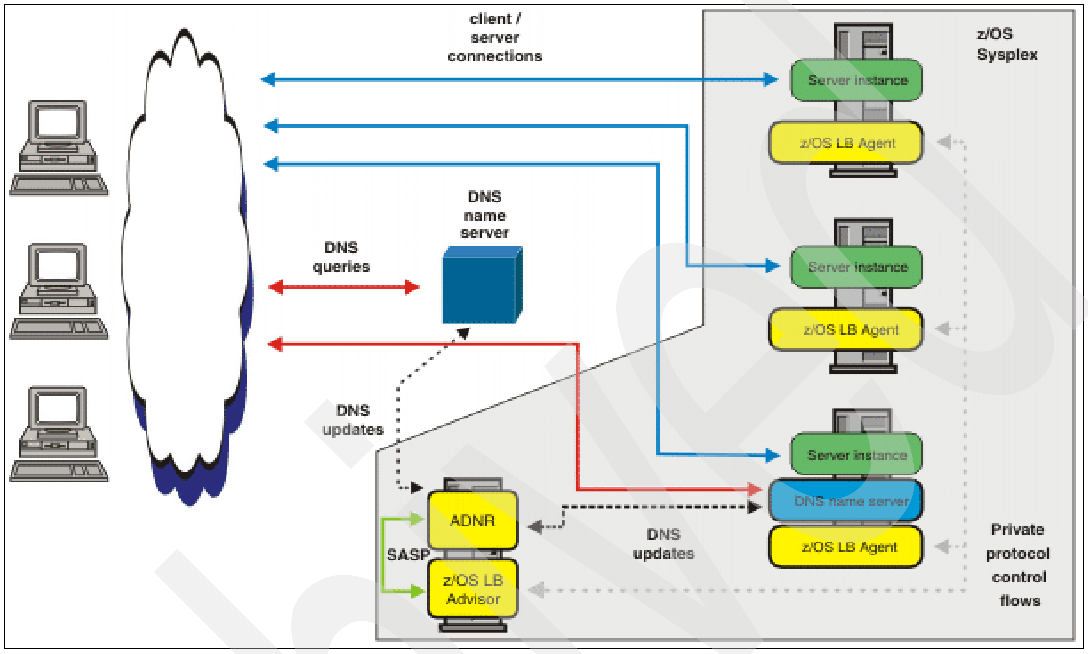


Figure 10-3 Diagram representing the flow among ADNR, Load Balancing Advisor, and agents.

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).

In Example 10-20 we note some basics elements of ADNR.

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                10.30.2.99..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              10.30.80.10..3860 # LBA lb_connection_v4 address
  host_connection_addr  10.30.1.230     # Local address
} # end of gwm
# --------------------------ok--------------------------------
 host_group              itso_sysplex                               1

  {
  host_group_name       SC3Xgroup                                   2
  dns                   dnsserv
  zone                 example.com

  member
  {
    host_name           sc31            # Prepended to domain suffix
    ipaddrlist          sc31_vipa_addrs
  }
  } # end of host_group definition
ipaddrlist              sc31_vipa_addrs
  {
    ipaddr                10.30.1.242     # static VIPA on sc31
  } # end of ipaddrlist
```

- **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.

The statement *member* defines a member for a given group. You should define the name of the individual host to be updated in the name server through the *host_name* parameter. The *ipaddrlist* parameter is the IP address of the host referenced before.

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 10-21.

Example 10-21 Example showing the convergence between ADNR and Load Balancing Advisor

```
F ADNR,DISPLAY,GWM
EZD1254I GWM SUMMARY 374
GWM LABEL        : Z/OS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
1 OF 1 RECORDS DISPLAYED
```

Once 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.

Important: 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.

You do not need to have Sysplex Distributor implemented in your environment to have ADNR configured.

Configuration

In our configuration we used a basic customization, including just one IP address to monitor, as shown in Example 10-20 on page 307. Basically, 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.

In order to have the configuration running and all the components active, we have the following recommendations:

– Configure your TCPIP.DATA to point to the DNS server that is updated by ADNR. In our configuration we used the parameters shown in Example 10-22.

Example 10-22 TCPIP.DATA file configuration

```
TCPIPJOBNAME TCPIPC
HOSTNAME WTSC30C
DOMAINORIGIN  EXAMPLE.COM  1
DATASETPREFIX TCPIPC
MESSAGECASE MIXED
NSINTERADDR  10.30.2.99    2
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
```

1 The domain should match with the zone defined on the DNS server.

2 The Name Server IP address must be the IP address of the DNS server.

– Example 10-23 on page 310 shows our DNS server configuration file (named.conf).

Note: 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.

Example 10-23 named.conf configuration file - DNS running on Linux

```
[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;
     file "example.com.zone";
     allow-update { any; };
     };
zone "2.30.10.in-addr.arpa" in {
     type master;
     file "2.30.10.zone";
     allow-update { any; };
     };
zone "1.30.10.in-addr.arpa" in {
     type master;
     notify no;
     file "1.30.10.zone";
     allow-update { any; };
     };
zone "0.0.127.in-addr.arpa" in {
     type master;
     file "localhost.zone";
     allow-update { none; };
     };
```

– Example 10-20 on page 307 is our starting point.

– You must configure the Load Balancing Advisor and the Agents, as shown in Example 10-24, Example 10-25 on page 311, and Example 10-26 on page 311.

Example 10-24 Load Balancing Advisor configuration file

```
#Load Balancing Advisor Configuration
debug_level              7
update_interval          60
agent_connection_port    8100

agent_id_list
{
  10.30.1.230..8000
  10.30.1.242..8000
}
```

```
lb_connection_v4          10.30.80.10..3860
lb_id_list
{
  10.30.2.1
  10.30.1.230
}
wlm serverwlm
port_list
{
  7000
  {
    wlm serverwlm
  }
}
```

Example 10-25 Load Balancing Agent configuration file for SC31

```
debug_level               7
advisor_id                10.30.80.10..8100
host_connection           10.30.1.242..8000
```

Example 10-26 Load Balancing Agent configuration file for SC30

```
debug_level               7
advisor_id                10.30.80.10..8100
host_connection           10.30.1.230..8000
```

Testing the ADNR

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:

1. Verify you can **ping** to the sysplex member DNS name (in our case: SC31.example.com)
2. Stop the TCP/IP stack in the sysplex member (in our case: TCPIPC on SC31)
3. Verify that the DNS status has changed for ADNR
4. Verify that you cannot **ping** the sysplex member
5. Start the TCP/IP stack
6. Verify that you can **ping** the sysplex member again

Verify you can ping the sysplex member DNS name

We issued the following command from SC30:

```
ping sc31.example.com
```

and received the output shown in Example 10-27.

Example 10-27 Ping from the SC30 system against the name registered on DNS server through ADNR

```
CS V1R8: Pinging host SC31.EXAMPLE.COM (10.30.1.242)
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 the 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 not present anymore (see Example 10-28).

Example 10-28 Display of the zones on DNS

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 936
DNS LABEL        : DNSSERV
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.30.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : EXAMPLE.COM.
  ZONE TIMESTAMP : 08/29/06 23:23:43
  TSIG FLAGS     :
  DNS RR LABEL   : SC31
   DNS RR STATUS : NOT_PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.30.1.242
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 08/30/06 14:34:51
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : NOT_PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.30.1.242
GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 08/30/06 14:34:51
1 OF 1 RECORDS DISPLAYED
```

The current status, NOT_PRESENT, indicates that there is no connectivity to SC31.

Verify that you cannot ping the sysplex member DNS name

Try a **ping** command again (from SC30 in our environment):

```
ping sc31.example.com
```

This produced the results shown in Example 10-29.

Example 10-29 Ping with no success after we stopped the TCPIPC on SC31.

```
ping sc31.example.com
EZZ3111I Unknown host 'SC31.EXAMPLE.COM'
```

Start the TCP/IP stack in the sysplex member

Start the TCP/IP stack on the sysplex member (in our case: SC31). This action will cause the IP address to become available. After starting the TCP/IP stack, issue the following command:

```
F ADNR,DISP,DNS,ZONES,DETAIL
```

You may initially see output similar that shown in Example 10-30. The process takes several seconds to update the DNS server and you might see *Update-Add_in_Progress* as intermediate status. If you issue the command again after a short time you should see the results similar to that shown in Example 10-31.

Example 10-30 Starting the TCP/IP stack and issuing display of the DNS RR status

```
S TCPIPC
$HASP100 TCPIPC   ON STCINRDR
IEF695I START TCPIPC   WITH JOBNAME TCPIPC   IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TCPIPC   STARTED
EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPC ARE AVAILABLE.

F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 965
DNS LABEL        : DNSSERV
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.30.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : EXAMPLE.COM.
  ZONE TIMESTAMP : 08/29/06 23:23:43
  TSIG FLAGS     :
  DNS RR LABEL   : SC31
   DNS RR STATUS : UPDATE-ADD_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.30.1.242
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 08/30/06 14:55:51
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : UPDATE-ADD_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.30.1.242
```

Example 10-31 DNS RR status changes to PRESENT

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 955
DNS LABEL        : DNSSERV
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.30.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : EXAMPLE.COM.
  ZONE TIMESTAMP : 08/29/06 23:23:43
  TSIG FLAGS     :
  DNS RR LABEL   : SC31
```

```
   DNS RR STATUS : PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.30.1.242
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 08/30/06 14:50:33
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
```

Problem determination

If there are problems with ADNR, try the following steps:

1. Verify that the DNS server, Load Balancer Advisor and ADNR started tasks are running
1. Verify that the GWM is active
2. Verify that members of the sysplex which are defined to ADNR are available
3. Verify that the DNS is active
4. More tests, in case the steps above have not succeeded.

Verifying 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.

Verifying that the GWM is active

Issue the following console command:

```
F ADNR,DISP,GWM,DETAIL
```

Example 10-32 Display the GWM status

```
F ADNR,DISP,GWM,DETAIL
EZD1254I GWM DETAIL 790
GWM LABEL        : Z/OS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
 GWM TIMESTAMP   : 08/29/06 15:00:06
 GWM IPADDR..PORT: 10.30.80.10..3860
 LOCAL IPADDR    : 10.30.1.230
 UUID            : IBM_SYSPLEX_ADNR
 UPDATE INTERVAL : 60
 LAST UPDATE     : N/A
1 OF 1 RECORDS DISPLAYED
F ADNR,DISP,GWM,GROUPS
EZD1254I GWM GROUP SUMMARY 792
GWM LABEL        : Z/OS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
 GROUP LABEL     : ITSO_SYSPLEX
  GROUP NAME     : SC3XGROUP
1 OF 1 RECORDS DISPLAYED
```

You may 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.

Verifying 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 10-33.

Example 10-33 Display of the GROUP members availability status

```
F ADNR,DISP,GWM,GROUPS,GROUPID=ITSO_SYSPLEX,DETAIL
EZD1254I GWM GROUP DETAIL 910
GWM LABEL        : ZOS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
 GWM TIMESTAMP   : 08/29/06 23:23:40
 GWM IPADDR..PORT: 10.30.80.10..3860
 LOCAL IPADDR    : 10.30.1.230
 UUID            : IBM_SYSPLEX_ADNR
 UPDATE INTERVAL : 60
 LAST UPDATE     : 08/30/06 13:38:03
 GROUP LABEL     : ITSO_SYSPLEX
  GROUP NAME     : SC3XGROUP
  GROUP TYPE     : HOST
  DNS LABEL      : DNSSERV
  ZONE LABEL     : MYZONE
  MEMBER HOSTNAME:
   IPADDR        : 10.30.1.242
    AVAIL        : YES
    FLAGS        :
    UPDATE COUNT : 3
  MEMBER HOSTNAME: SC31
   IPADDR        : 10.30.1.242
    AVAIL        : YES
    FLAGS        :
    UPDATE COUNT : 3
1 OF 1 RECORDS DISPLAYED
```

The name is registered with the DNS server is sc31.example.com, where the first part of the name (sc31) came from the member hostname. The remainder of the name is from domain_suffix defined on the zone statement as shown in Example 10-20 on page 307.

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, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Verifying that the DNS status is active

Issue the following console command:

```
F ADNR,DISP,DNS,DETAIL
```

The results are similar to Example 10-34. 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 10-34 Displaying the DNS status

```
F ADNR,DISP,DNS,DETAIL
EZD1254I DNS DETAIL 915
DNS LABEL        : DNSSERV
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.30.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
1 OF 1 RECORDS DISPLAYED
```

More tests

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 USS.

   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 very 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. You may do it by updating the DNS configuration file (usually it will be /etc/named.conf) to include the following line, as shown in Example 10-35.

Example 10-35 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 may 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:

   ```
   nslookup
   ```

   And from the nslookup program issue the following command:

   ```
   server 10.30.2.99
   ```

   Where 30.10.2.99 represents the DNS server IP address. You are supposed to get the following screen:

   ```
      Default Server:  dnsserv.example.com
      Address:  10.30.2.99
   >
   ```

   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 10-36.

Example 10-36 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 10-37 in the ADNR log file.

Example 10-37 Error message in the ADNR log file

```
Zone transfer for zone myzone failed, rc = No file could be created
```

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. The *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, refer to the *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 321 show sample commands and applications that use the different 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 321. The variables associated with the Language Environment API are listed in Table A-4 on page 322.

The variables specific to each application are listed in Table A-5 on page 323. 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" on page 320
- "z/OS UNIX API environment" on page 321
- "z/OS UNIX System Services environment variables" on page 321
- "Language Environment, environment variables" on page 322
- "Application-specific environment variables" on page 323
- "Setting environment variables" on page 329

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 some 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
- Unix System Services

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, 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

LE Environment variable	Description
_CEE_DLLLOAD_XPCOMPAT	Controls z/OS V1R6 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 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

The application-specific environment variables are listed in Table A-5.

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 321. 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 322. 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 via 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 via program call
Language Environment API variables	Has indirect access via 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 321

Language Environment API variables	Has access to the variables in Table A-4 on page 322
IKED	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
IKED_FILE	Configuration file for IKED
IKED_CTRACE_MEMBER	Parmlib member containing CTRACE settings for IKED
Load Balancing Advisor and Agent	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
TN3270E Telnet server	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 321
Language Environment API variables	No access to the variables in Table A-4 on page 322
INETD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
OTELNETD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
FTP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
KRB5_SERVER_KETAB	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 321

Language Environment API variables	Has access to the variables in Table A-4 on page 322
TSO REXEC	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 321
Language Environment API variables	No access to the variables in Table A-4 on page 322
UNIX REXECD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
UNIX RSHD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
_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 321
Language Environment API variables	No access to the variables in Table A-4 on page 322

Sendmail	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	No access to the variables in Table A-4 on page 322
DHCP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
LPR_PRINTER	Output printer for DHCP reports
BIND4	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
SNTPD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
PORTMAPPER	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
UNIX PORTMAPPER	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
MISCSERV	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322

DCAS	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 321
Language Environment API variables	Has access to the variables in Table A-4 on page 322
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 the HFS file /etc/tcpa.data, you can code the following export command:

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

If instead of an HFS file, you want to set RESOLVER_CONFIG to the data set MVSA.PROD.PARMS(TCPDATA), you can specify the following export command. Be 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 needs to be passed as a parameter in the JCL for the application. For example, the following shows the RESOLVER_CONFIG variable pointing to an HFS 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(TCPDATA),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. 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 an HFS 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 V1R7.0 Communications Server product. These samples can be found in either of two locations:

- SYS1.SEZAINST, an MVS PDS
- /usr/lpp/tcpip/samples/, an HFS directory

Sample files by component

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 verification procedure for Telnet
/usr/lpp/tcpip/samples/mvstn3270.mi2	mibs for TN3270
OTELNETD	
/usr/lpp/tcpip/samples/services	/etc/services file
/usr.lpp/tcpip/samples/inetd.conf	/etc/inetd.conf file
INETD	
/usr/lpp/tcpip/samples/services	/etc/services file
/usr.lpp/tcpip/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
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
BIND4	
SEZAINST(NAMED4)	JCL for BIND 4 Proc
/usr/lpp/tcpip/samples/named.boot	Boot file for bind4
/usr/lpp/tcpip/samples/db.mycorp.v4	Sample version 4 file
/usr/lpp/tcpip/samples/db.34.37.9.v4	Sample version 4 file
/usr/lpp/tcpip/samples/db.loopback.v4	Sample version 4 file
/usr/lpp/tcpip/samples/slave.boot	Boot file for a slave server
/usr/lpp/tcpip/samples/caching.boot	Boot file for a caching server
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 slave server
/usr/lpp/tcpip/samples/caching.conf	Boot file for a caching server
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
/usr.lpp/tcpip/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
/usr.lpp/tcpip/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(EZADCASP)	Alias JCL for DCAS proc
SEZAINST(DCAS)	JCL for DCAS proc
Load Balancing Advisor and Agent	
SEZAINST(EZBLBADV)	JCL for Advisor proc
SEZAINST(EZBLBADC)	JCL for Advisor configuration file
SEZAINST(EZBLBAGE)	JCL for Agent proc
SEZAINST(EZBLBAGC)	JCL for Agent configuration file
SEZAINST(LBADVCNF)	LBADV.CONF
SEZAINST(LBAGECNF)	LBAGENT.CONF

SLAP	
SEZAINST(EZAPAGSB)	Alias JCL for SLAPM2 subagent proc
SEZAINST(NSLAPM2)	JCL for SLAPM2 subagent proc
SEZAINST(EZAPAGSN)	Alias JCL for SLAP subagent proc
SEZAINST(PAGTSNMP)	JCL for SLAP 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(EZAPAGSP)	Alias JCL for Policy Agent proc
SEZAINST(PAGENT)	JCL for Policy Agent proc
Policy Agent Configuration	
/usr/lpp/tcpip/samples/pagent.conf	Configuration file for Policy Agent
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 2	
/usr/lpp/tcpip/samples/pagent_oc.conf	Object class definitions
/usr/lpp/tcpip/samples/pagent_at.conf	attribute Definitions
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 standalone started task scenario

This appendix contains the configuration files used for the TN3270E Telnet server standalone started task scenario.

SC30 files for TN3270 standalone started task scenario

The following sections are included:

- "SC30 TN3270B Server PROC for TN3270 standalone started task" on page 340
- "SC30 TN3270B Server PROFILE for TN3270 standalone started task" on page 340
- "SC30 TCPIPB stack PROC for TN3270 standalone started task" on page 342
- "SC30 TCPIPB stack PROFILE for TN3270 standalone started task" on page 343
- "SC30 OMPROUTE PROC for TN3270 standalone started task" on page 346
- "SC30 OMPROUTE STDENV file for TN3270 Standalone Task Scenario" on page 346
- "SC30 OMPROUTE CONFIG for TN3270 standalone started task" on page 347

SC30 TN3270B Server PROC for TN3270 standalone started task

Figure C-1 lists the JCL procedure for the SC30 TN3270B server.

```
 BROWSE    SYS1.PROCLIB(TN3270B) - 01.02             Line 00000000 Co
 Command ===>                                                Scroll
********************************* Top of Data ***********************
//TN3270B  PROC PARMS='TRC=TN',
//             PROFILE=TELNB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//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.)
******************************** Bottom of Data *********************
```

Figure C-1 SC30 TN3270B PROC JCL

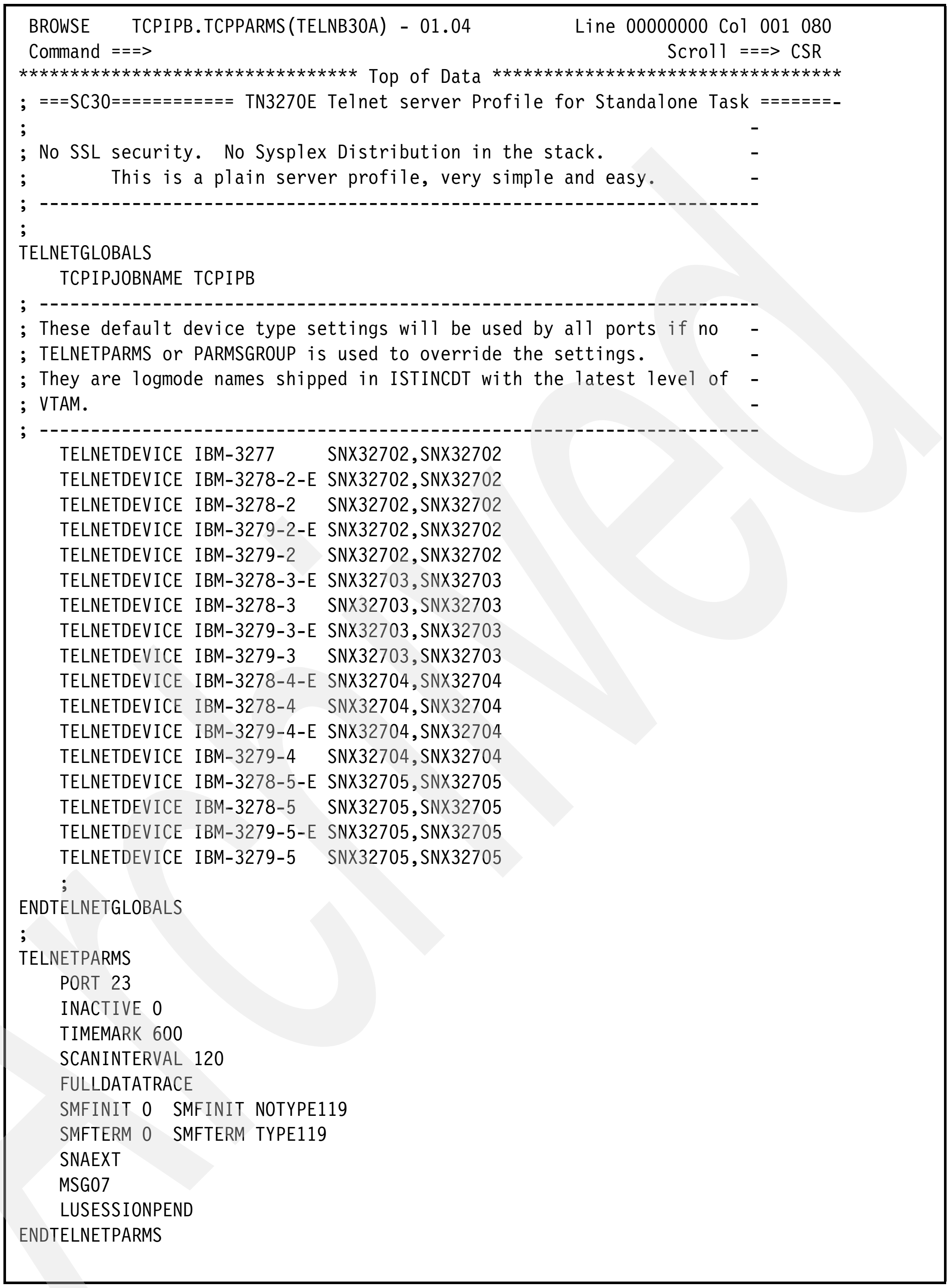
We started the server task by issuing the following command:

```
S TN3270B,PROFILE=TELNB30A
```

SC30 TN3270B Server PROFILE for TN3270 standalone started task

The profile for the TN3270E Telnet server for this scenario is TELNB30A. 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).



```
 BROWSE    TCPIPB.TCPPARMS(TELNB30A) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
********************************* Top of Data **********************************
; ===SC30============ TN3270E Telnet server Profile for Standalone Task =======-
;                                                                        -
; No SSL security.  No Sysplex Distribution in the stack.               -
;       This is a plain server profile, very simple and easy.           -
; ----------------------------------------------------------------------
;
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   SC30 TN3270B profile (TELNB30A)

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS

  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
******************************** Bottom of Data ********************************
```

Figure C-3 SC30 TN3270B profile (TELNB30A) - end

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC30BB*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 SC30BBnn.

SC30 TCPIPB stack PROC for TN3270 standalone started task

The JCL procedure for the stack is TCPIPB.

```
//TCPIPB    PROC PARMS='CTRACE(CTIEZBOO),IDS=00',
//             PROFILE=PROFB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE
//TCPIPB    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM=('&PARMS',
//          'ENVAR("RESOLVER_CONFIG=//''TCPIPB.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=*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*TNDBCSER DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure C-4 SC30 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB30A
```

SC30 TCPIPB stack PROFILE for TN3270 standalone started task

The PROFILE for the stack is PROFB30A, as shown in Figure C-5, Figure C-6 on page 344, Figure C-7 on page 345, and Figure C-8 on page 346.

```
 BROWSE    TCPIPB.TCPPARMS(PROFB30A) - 01.06          Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
********************************* Top of Data **********************************
;
; TCPIP.PROFILE.TCPIP     for Standalone TN3270 and no Sysplex Dist.
; ==================
;
GLOBALCONFIG
   ECSALIMIT OM              ; default = OM
   POOLLIMIT OM              ; default = OM
   TCPIPSTATISTICS           ; default = notcpipstatistics
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
   RESTRICTLOWPORTS          ; default = restrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   DELAYACKS                 ; default = delayacks
;
UDPCONFIG
   RESTRICTLOWPORTS          ; default = restrictlowports
   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 C-5   SC30 TCPIPB profile (PROFB30A)

```
; ----------------------------------------------------------------------
 AUTOLOG 5

   OMPB

 ENDAUTOLOG
; ----------------------------------------------------------------------

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    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                         ; control port
    23 TCP OMVS    BIND 10.20.10.241     ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG            ; MVS Telnet Server
    25 TCP SMTPB                        ; SMTP Server
    53 TCP NAMED9B                      ; Domain Name Server
    53 UDP NAMED9B                      ; Domain Name Server
   111 TCP PORTMAPB                     ; Portmap Server
   111 UDP PORTMAPB                     ; 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 10.20.10.242       ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.20.10.242       ; 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
   750 TCP MVSKERBB                     ; Kerberos
   750 UDP MVSKERBB                     ; Kerberos
   751 TCP ADM@SRVB                     ; Kerberos Admin Server
   751 UDP ADM@SRVB                     ; Kerberos Admin Server
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
```

Figure C-6   SC30 TCPIPB profile (PROFB30A) - continued

```
;--------------------------------------------------------------
;Hipersocks: trle definitions are dynamically created on vtam  -
;--------------------------------------------------------------
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4LNK  IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5LNK  IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6LNK  IPAQIDIO      IUTIQDF6
;
;--------------------------------------------------------------
; Static VIPA definitions                                      -
;--------------------------------------------------------------
 DEVICE STAVIPA1    VIRTUAL 0
 LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
;
;--------------------------------------------------------------
;Osa definitions... relates to sys1.vtamlst trle definitions   -
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0             -
;--------------------------------------------------------------
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 20
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0LNK  IPAQENET      OSA20A0 VLANID 21
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0LNK  IPAQENET      OSA20C0 VLANID 20
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 21
;
HOME
   10.20.4.234   IUTIQDF4LNK
   10.20.4.235   IUTIQDF5LNK
   10.20.5.236   IUTIQDF6LNK
   10.20.1.230   STAVIPA1LNK
   10.20.2.232    OSA2080LNK
   10.20.3.233    OSA20A0LNK
   10.20.2.234    OSA20C0LNK
   10.20.3.235    OSA20E0LNK
;
  PRIMARYINTERFACE STAVIPA1LNK

 VIPADYNAMIC
  ;-----------------------------------------------------------------
  ;  Set aside 14 addresses for use with BIND and SIOCSVIPA IOCTL    -
  ;             (10.20.10.241 thru .254)                             -
  ;-----------------------------------------------------------------
   VIPARANGE  DEFINE  255.255.255.240  10.20.10.240

 ENDVIPADYNAMIC
```

Figure C-7   SC30 TCPIPB profile (PROFB30A) - continued

```
ITRACE OFF

START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
START OSA2080
START OSA20A0
START OSA20C0
START OSA20E0
;
******************************** Bottom of Data ********************************
```

Figure C-8   SC30 TCPIPB profile (PROFB30A) - end

SC30 OMPROUTE PROC for TN3270 standalone started task

Figure C-9 shows the JCL procedure for OMPROUTE.

```
***************************** Top of Data ********************
//OMPB   PROC STDENV=STDNVB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*STDOUT   DD PATH='/etc/omproute/omproute.stdout',
//*             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
**************************** Bottom of Data *******************
```

Figure C-9   SC30 OMPROUTE PROC (OMPB)

SC30 OMPROUTE STDENV file for TN3270 Standalone Task Scenario

Figure C-10 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(STDNVB30) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
********************************* Top of Data **********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB30)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB30)'
OMPROUTE_DEBUG_FILE=/etc/omproute/debug30b
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure C-10   SC30 OMPROUTE STDENV file (STDNVB30)

SC30 OMPROUTE CONFIG for TN3270 standalone started task

Figure C-11 shows the config for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB30) - 01.00           Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
******************************** Top of Data **********************************
Area Area_Number=0.0.0.0
     Stub_Area=NO
     Authentication_type=None
     Import_Summaries=No;
;
OSPF RouterID=10.20.1.230;
Global_Options Ignore_Undefined_Interfaces=yes;
;
Routesa_Config Enabled=No;
;
; hipersocks  IUTIQDF4LNK
 ospf_interface ip_address=10.20.4.234
               subnet_mask=255.255.255.0
               name=IUTIQDF4LNK
               attaches_to_area=0.0.0.0
               cost0=10;
;
; hipersocks  IUTIQDF5LNK
 ospf_interface ip_address=10.20.4.235
               subnet_mask=255.255.255.0
               name=IUTIQDF5LNK
               ROUTER_PRIORITY=0
               attaches_to_area=0.0.0.0
               cost0=10;
;
; hipersocks  IUTIQDF6LNK
 ospf_interface ip_address=10.20.5.236
               subnet_mask=255.255.255.0
               name=IUTIQDF6LNK
               attaches_to_area=0.0.0.0
               cost0=10;
;
; ************************************************************
; *Static 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.
; ************************************************************
ospf_interface ip_address=10.20.1.230
          subnet_mask=255.255.255.252
          destination_addr=10.20.1.230
          name=STAVIPA1LNK
          Advertise_VIPA_Routes=HOST_ONLY
          Router_Priority=0
          attaches_to_area=0.0.0.0
          cost0=10
          mtu=1500;
```

Figure C-11 SC30 OMPROUTE config (OMPB30)

```
; OSA Qdio OSA2080LNK
ospf_interface ip_address=10.20.2.232
           subnet_mask=255.255.255.0
           name=OSA2080LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20A0LNK
ospf_interface ip_address=10.20.3.233
           subnet_mask=255.255.255.0
           name=OSA20A0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20C0LNK
ospf_interface ip_address=10.20.2.234
           subnet_mask=255.255.255.0
           name=OSA20C0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
          mtu=1500;
;
; OSA Qdio OSA20E0LNK
ospf_interface ip_address=10.20.3.235
           subnet_mask=255.255.255.0
           name=OSA20E0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; ***********************************************************
; *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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ***********************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
```

Figure C-12   SC30 OMPROUTE config (OMPB30) - continued

```
;
; ************************************************************
; *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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* 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=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure C-13   SC30 OMPROUTE Config (OMPB30) - end

D

Configuration files: TN3270E Telnet server with connection security scenario

This appendix contains the configuration files used for the TN3270E Telnet server with the connection security features scenario.

SC30 files for TN3270E Telnet server with connection security scenario

The following sections are included:

- "SC30 TN3270B Server PROC for TN3270 with connection security" on page 352
- "SC30 TN3270B Server PROFILE for TN3270 with connection security" on page 352
- "SC30 TCPIPB stack PROC for TN3270 with connection security" on page 356
- "SC30 TCPIPB stack PROFILE for TN3270 with connection security" on page 356
- "SC30 OMPROUTE PROC for TN3270 with connection security" on page 360
- "SC30 OMPROUTE STDENV file for TN3270 with connection security" on page 360
- "SC30 OMPROUTE CONFIG for TN3270 with connection security" on page 360

SC30 TN3270B Server PROC for TN3270 with connection security

Figure D-1 shows the JCL procedure for the TN3270E Telnet server.

```
 BROWSE    SYS1.PROCLIB(TN3270B) - 01.02             Line 00000000 Co
 Command ===>                                                 Scroll
********************************* Top of Data ************************
//TN3270B  PROC PARMS='TRC=TN',
//             PROFILE=TELNB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//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.)
******************************** Bottom of Data **********************
```

Figure D-1 SC30 TN3270B PROC JCL

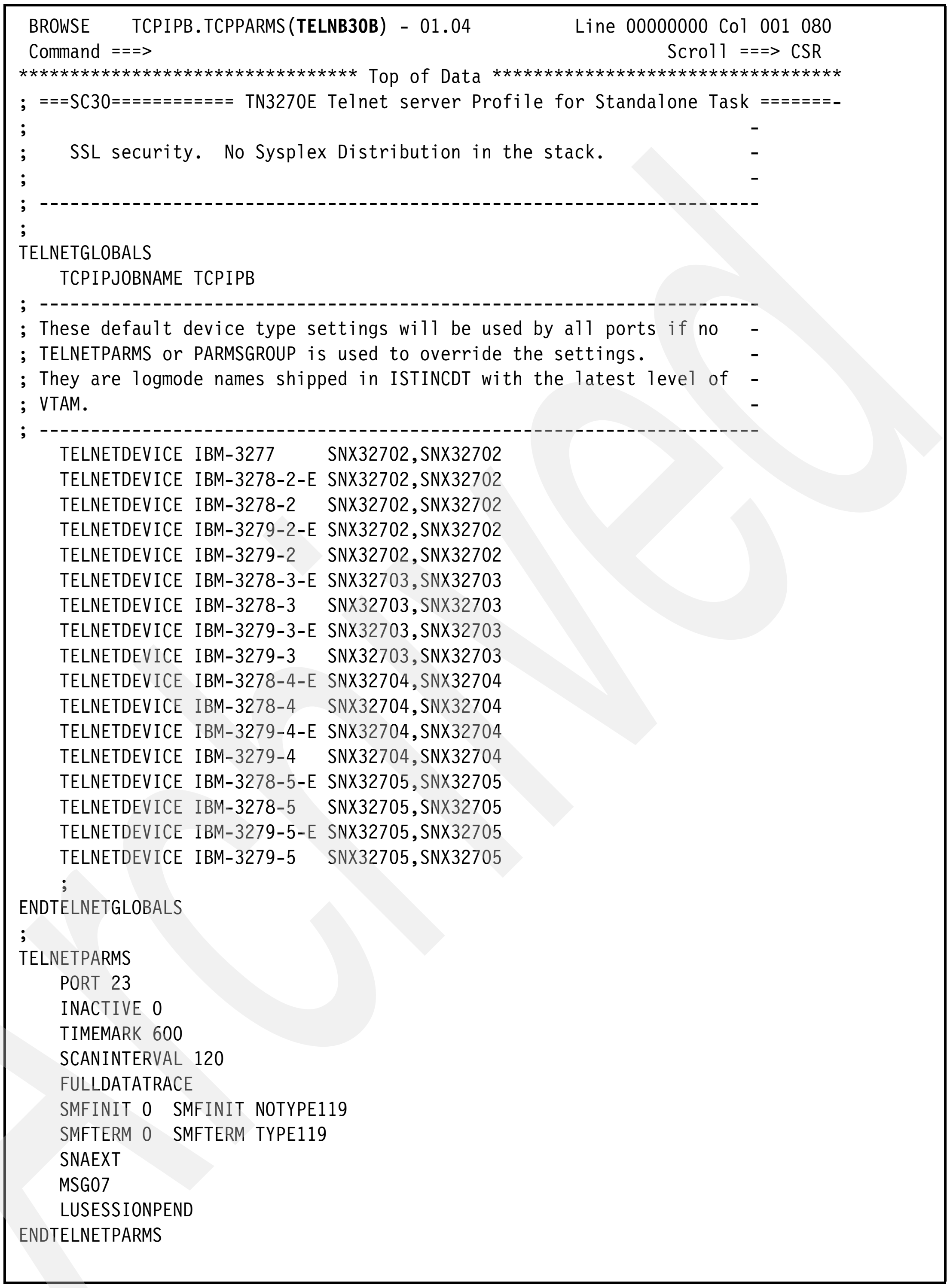
We started the server task for this scenario by issuing the following command:

```
S TN3270B,PROFILE=TELNB30B
```

SC30 TN3270B Server PROFILE for TN3270 with connection security

The profile for the TN3270E Telnet server for this scenario is TELNB30B, shown in Figure D-2 on page 353, Figure D-3 on page 354, and Figure D-4 on page 355. 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).



```
 BROWSE    TCPIPB.TCPPARMS(TELNB30B) - 01.04          Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
******************************** Top of Data **********************************
; ===SC30=========== TN3270E Telnet server Profile for Standalone Task =======-
;                                                                         -
;    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 D-2   SC30 TN3270B profile (TELNB30B)

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS

  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
;
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING HFS /etc/sc30b.keyring.kdb     ;keyring used by all secure
    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
```

Figure D-3 SC30 TN3270B profile (TELNB30B) - continued

```
; ---------------------------------------------------------------------
; 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

  DESTIPGROUP GENERALUSER 10.20.10.21   ENDDESTIPGROUP
  DESTIPGROUP ADMIN       10.20.10.22   ENDDESTIPGROUP
  DESTIPGROUP PAYROLL     10.20.10.23   ENDDESTIPGROUP
  DESTIPGROUP SHIPPING    10.20.1.230   ENDDESTIPGROUP
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLP0  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER      DESTIPGRP,SHIPPING
  DEFAULTAPPL TSO       DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE
 ;-----------------------------------------------------------------
 ; 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.                    -
 ;-----------------------------------------------------------------
  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
;
******************************** Bottom of Data ********************************
```

Figure D-4   SC30 TN3270B profile (TELNB30B) - end

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC30BB*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 decided to use 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.

SC30 TCPIPB stack PROC for TN3270 with connection security

The JCL procedure for the stack is TCPIPB, listed in Figure D-5.

```
//TCPIPB    PROC PARMS='CTRACE(CTIEZB00),IDS=00',
//             PROFILE=PROFB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE
//TCPIPB    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM=('&PARMS',
//         'ENVAR("RESOLVER_CONFIG=//''TCPIPB.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=*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*TNDBCSER DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

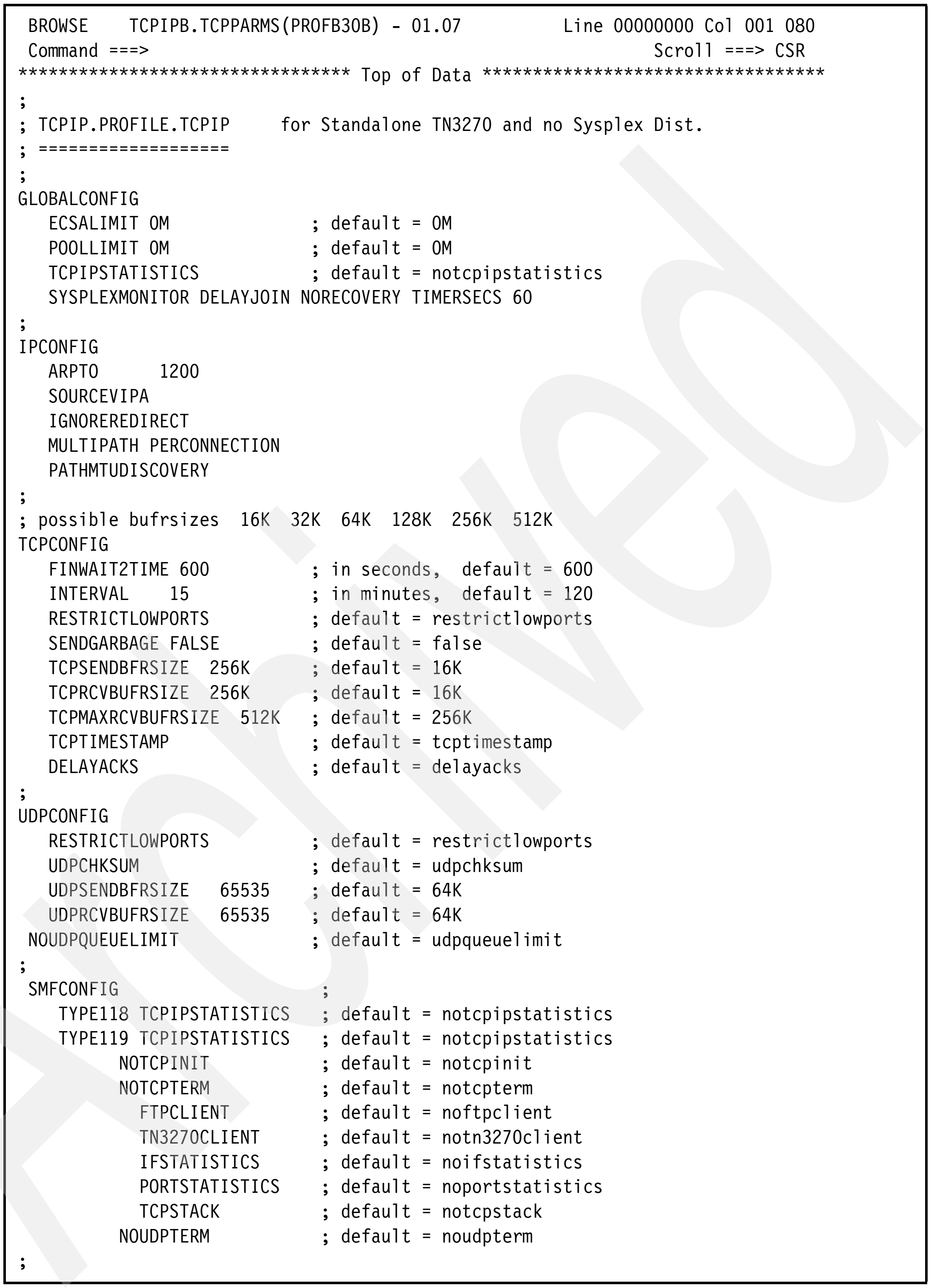
Figure D-5 SC30 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB30B
```

SC30 TCPIPB stack PROFILE for TN3270 with connection security

The PROFILE for the stack is PROFB30B, listed in Figure D-6 on page 357, Figure D-7 on page 358, Figure D-8 on page 359, and Figure D-9 on page 360.



```
 BROWSE    TCPIPB.TCPPARMS(PROFB30B) - 01.07          Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************* Top of Data **********************************
;
; TCPIP.PROFILE.TCPIP     for Standalone TN3270 and no Sysplex Dist.
; ==================
;
GLOBALCONFIG
   ECSALIMIT OM              ; default = OM
   POOLLIMIT OM              ; default = OM
   TCPIPSTATISTICS           ; default = notcpipstatistics
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
   RESTRICTLOWPORTS          ; default = restrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   DELAYACKS                 ; default = delayacks
;
UDPCONFIG
   RESTRICTLOWPORTS          ; default = restrictlowports
   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-6   SC30 TCPIPB profile (PROFB30B)

```
; ----------------------------------------------------------------------
 AUTOLOG 5

   OMPB

 ENDAUTOLOG
; ----------------------------------------------------------------------

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    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.20.10.241     ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server
    25 TCP SMTPB                         ; SMTP Server
    53 TCP NAMED9B                       ; Domain Name Server
    53 UDP NAMED9B                       ; Domain Name Server
   111 TCP PORTMAPB                      ; Portmap Server
   111 UDP PORTMAPB                      ; 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 10.20.10.242        ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.20.10.242        ; UNIX RSHD   D-VIPA
   512 TCP RXSERVEB                      ; TSO  REXECD
   514 TCP RXSERVEB                      ; TSO  RSHD
   515 TCP LPSERVEB                      ; LPD Server
   520 UDP OMPB    NOAUTLOG              ; OMPROUTE
;  580 UDP NCPROUT             ; NCPROUTE Server
   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             ; MVS Telnet Server SECURE
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
;----------------------------------------------------------------
;Hipersocks: trle definitions are dynamically created on vtam  -
;----------------------------------------------------------------
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4LNK  IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5LNK  IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6LNK  IPAQIDIO      IUTIQDF6
```

Figure D-7 SC30 TCPIPB profile (PROFB30B) - continued

```
;
;---------------------------------------------------------------
; Static VIPA definitions                                      -
;---------------------------------------------------------------
 DEVICE STAVIPA1    VIRTUAL 0
 LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
;
;---------------------------------------------------------------
;Osa definitions... relates to sys1.vtamlst trle definitions   -
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0             -
;---------------------------------------------------------------
DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 20
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0LNK  IPAQENET      OSA20A0 VLANID 21
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0LNK  IPAQENET      OSA20C0 VLANID 20
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 21
;
HOME
   10.20.4.234   IUTIQDF4LNK
   10.20.4.235   IUTIQDF5LNK
   10.20.5.236   IUTIQDF6LNK
   10.20.1.230   STAVIPA1LNK
   10.20.2.232    OSA2080LNK
   10.20.3.233    OSA20A0LNK
   10.20.2.234    OSA20C0LNK
   10.20.3.235    OSA20E0LNK
;
  PRIMARYINTERFACE STAVIPA1LNK

 VIPADYNAMIC
  ;------------------------------------------------------------------
  ;  Set aside 14 addresses for use with BIND and SIOCSVIPA IOCTL    -
  ;             (10.20.10.241 thru .254)                             -
  ;------------------------------------------------------------------
   VIPARANGE  DEFINE  255.255.255.240  10.20.10.240

  ;------------------------------------------------------------------
  ;  Define addresses to be used with TN3270E Telnet server and SSL         -
  ;             (10.20.10.21  thru .23)                              -
  ;------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payroll

 ENDVIPADYNAMIC

ITRACE OFF
```
Figure D-8 SC30 TCPIPB profile (PROFB30B) - continued

```
START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
START OSA2080
START OSA20A0
START OSA20C0
START OSA20E0
;
******************************** Bottom of Data ********************************
```

Figure D-9 SC30 TCPIPB profile (PROFB30B) - end

SC30 OMPROUTE PROC for TN3270 with connection security

Figure D-10 shows the JCL procedure for OMPROUTE.

```
***************************** Top of Data ********************
//OMPB   PROC STDENV=STDNVB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*STDOUT   DD PATH='/etc/omproute/omproute.stdout',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
**************************** Bottom of Data *******************
```

Figure D-10 SC30 OMPROUTE PROC (OMPB)

SC30 OMPROUTE STDENV file for TN3270 with connection security

Figure D-11 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(STDNVB30) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
********************************* Top of Data **********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB30)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB30)'
OMPROUTE_DEBUG_FILE=/etc/omproute/debug30b
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure D-11 SC30 OMPROUTE STDENV file (STDNVB30)

SC30 OMPROUTE CONFIG for TN3270 with connection security

Figure D-12 on page 361, Figure D-13 on page 362, and Figure D-14 on page 363 lists the configuration for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB30) - 01.00            Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************* Top of Data **********************************
Area Area_Number=0.0.0.0
     Stub_Area=NO
     Authentication_type=None
     Import_Summaries=No;
;
OSPF RouterID=10.20.1.230;
Global_Options Ignore_Undefined_Interfaces=yes;
;
Routesa_Config Enabled=No;
;
; hipersocks  IUTIQDF4LNK
 ospf_interface ip_address=10.20.4.234
                subnet_mask=255.255.255.0
                name=IUTIQDF4LNK
                attaches_to_area=0.0.0.0
                cost0=10;
;
; hipersocks  IUTIQDF5LNK
 ospf_interface ip_address=10.20.4.235
                subnet_mask=255.255.255.0
                name=IUTIQDF5LNK
                ROUTER_PRIORITY=0
                attaches_to_area=0.0.0.0
                cost0=10;
;
; hipersocks  IUTIQDF6LNK
 ospf_interface ip_address=10.20.5.236
                subnet_mask=255.255.255.0
                name=IUTIQDF6LNK
                attaches_to_area=0.0.0.0
                cost0=10;
;
; ************************************************************
; *Static 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.
; ************************************************************
ospf_interface ip_address=10.20.1.230
           subnet_mask=255.255.255.252
           destination_addr=10.20.1.230
           name=STAVIPA1LNK
           Advertise_VIPA_Routes=HOST_ONLY
           Router_Priority=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
```

Figure D-12   SC30 OMPROUTE Config (OMPB30)

```
; OSA Qdio OSA2080LNK
ospf_interface ip_address=10.20.2.232
           subnet_mask=255.255.255.0
           name=OSA2080LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20A0LNK
ospf_interface ip_address=10.20.3.233
           subnet_mask=255.255.255.0
           name=OSA20A0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20C0LNK
ospf_interface ip_address=10.20.2.234
           subnet_mask=255.255.255.0
           name=OSA20C0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
          mtu=1500;
;
; OSA Qdio OSA20E0LNK
ospf_interface ip_address=10.20.3.235
           subnet_mask=255.255.255.0
           name=OSA20E0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; ************************************************************
; *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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
```

Figure D-13 SC30 OMPROUTE Config (OMPB30) - continued

```
;
; ************************************************************
; *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 therfore 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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* 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=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure D-14 SC30 OMPROUTE Config (OMPB30) - end

E

Configuration files: Multiple TN3270E Telnet servers and SD scenario

This appendix contains the configuration files used for the multiple TN3270E Telnet servers with the sysplex distribution scenario. Sysplex distribution is abbreviated as SD.

There are two sections in this appendix:

- “SC30 files for multiple TN3270E Telnet servers and SD scenario” on page 366
- “SC31 files for multiple TN3270E Telnet servers and SD scenario” on page 378

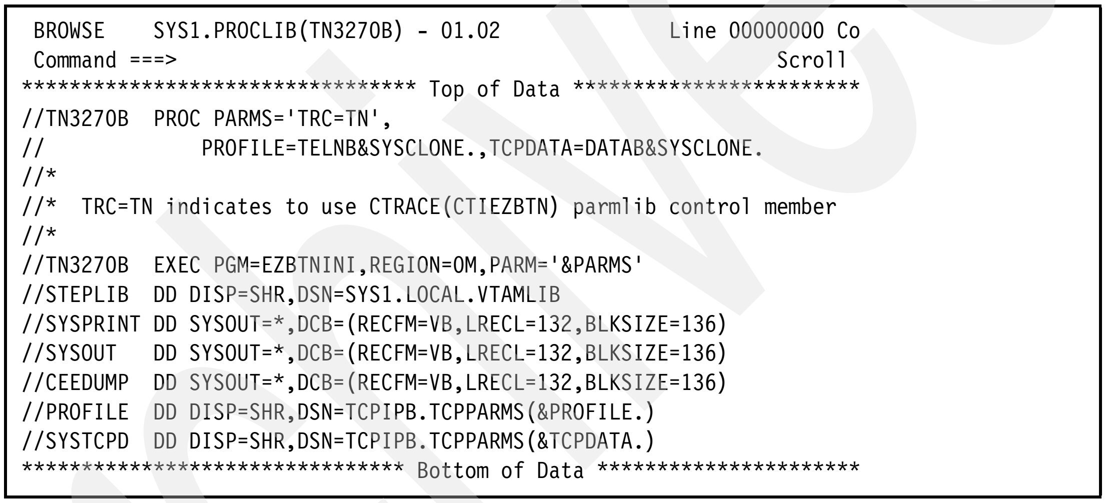
SC30 files for multiple TN3270E Telnet servers and SD scenario

The following sections are included:

- "SC30 TN3270B Server PROC for multiple TN3270E Telnet servers and SD" on page 366
- "SC30 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD" on page 366
- "SC30 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD" on page 370
- "SC30 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD" on page 370
- "SC30 OMPROUTE PROC for multiple TN3270E Telnet servers and SD" on page 375
- "SC30 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD" on page 375
- "SC30 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD" on page 376

SC30 TN3270B Server PROC for multiple TN3270E Telnet servers and SD

Figure E-1 shows the PROC JCL for the TN3270E Telnet server.



```
 BROWSE    SYS1.PROCLIB(TN3270B) - 01.02              Line 00000000 Co
 Command ===>                                                  Scroll
********************************* Top of Data ************************
//TN3270B  PROC PARMS='TRC=TN',
//             PROFILE=TELNB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//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.)
******************************** Bottom of Data **********************
```

Figure E-1 SC30 TN3270B PROC JCL

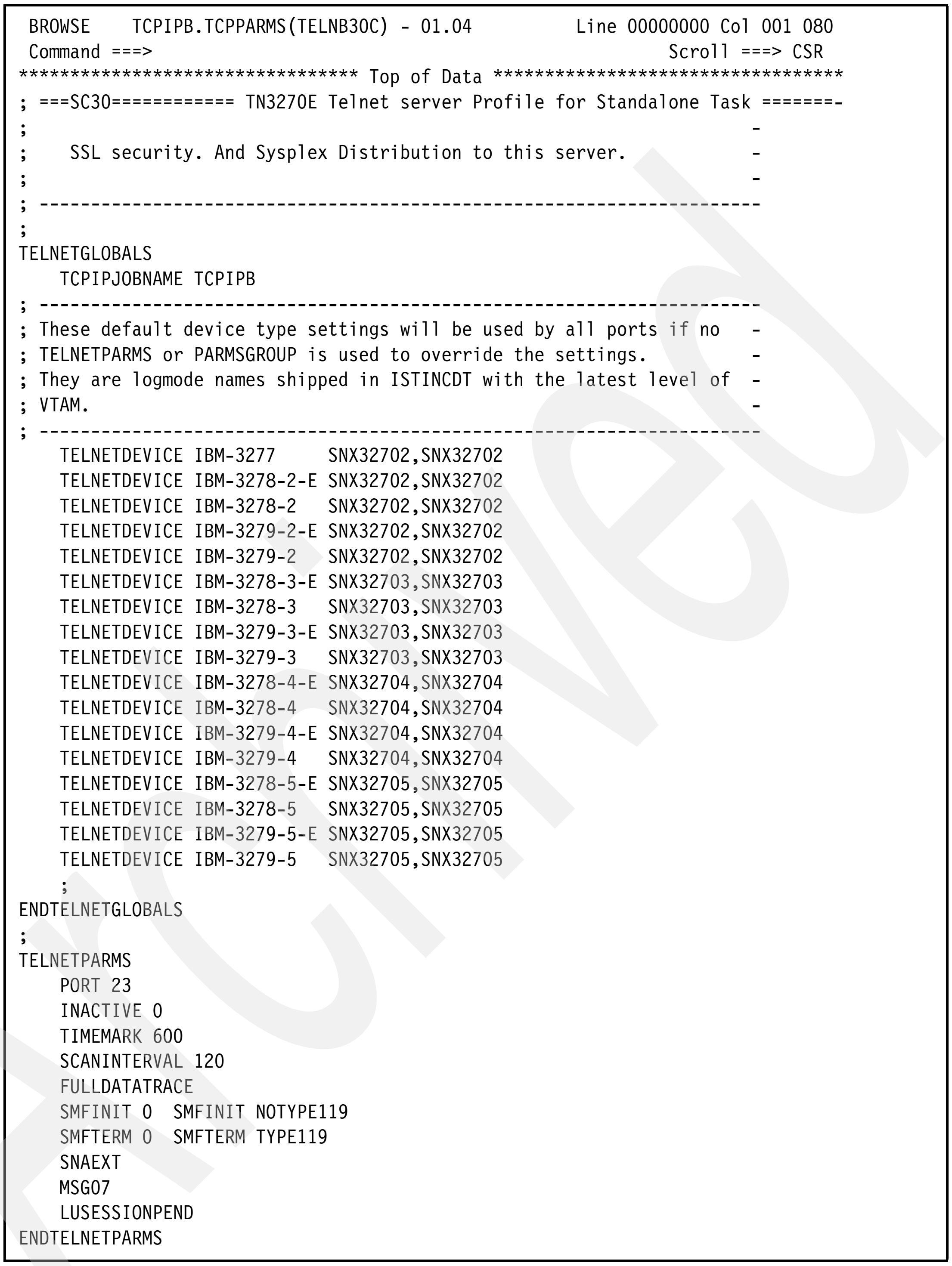
We started the server task for this scenario by issuing the following command:

```
S TN3270B,PROFILE=TELNB30C
```

SC30 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD

The profile for the TN3270E Telnet server for this scenario is TELNB30C. 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).



```
 BROWSE    TCPIPB.TCPPARMS(TELNB30C) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
********************************* Top of Data **********************************
; ===SC30============ TN3270E Telnet server Profile for Standalone Task =======-
;                                                                      -
;    SSL security. And Sysplex Distribution to this server.            -
;                                                                      -
; ----------------------------------------------------------------------
;
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 E-2 SC30 TN3270B profile (TELNB30C) - continued

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS

  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
;
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING HFS /usr/keyring/tcpcs.kdb     ;keyring used by all secure
    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 from the appl   -
; ---------------------------------------------------------------------
  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
```

Figure E-3 SC30 TN3270B profile (TELNB30C) - continued

```
; ---------------------------------------------------------------------
; 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

  DESTIPGROUP GENERALUSER 10.20.10.21   ENDDESTIPGROUP
  DESTIPGROUP ADMIN       10.20.10.22   ENDDESTIPGROUP
  DESTIPGROUP PAYROLL     10.20.10.23   ENDDESTIPGROUP
  DESTIPGROUP SHIPPING    10.20.1.230   ENDDESTIPGROUP
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLP0  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER      DESTIPGRP,SHIPPING
  DEFAULTAPPL TSO       DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE
 ;-----------------------------------------------------------------
 ; 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 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
;
******************************** Bottom of Data ********************************
```

Figure E-4   SC30 TN3270B profile (TELNB30C) - end

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC30B*?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.
- *?* 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 decided to use 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.

SC30 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD

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("RESOLVER_CONFIG=//''TCPIPB.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=*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*TNDBCSER DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

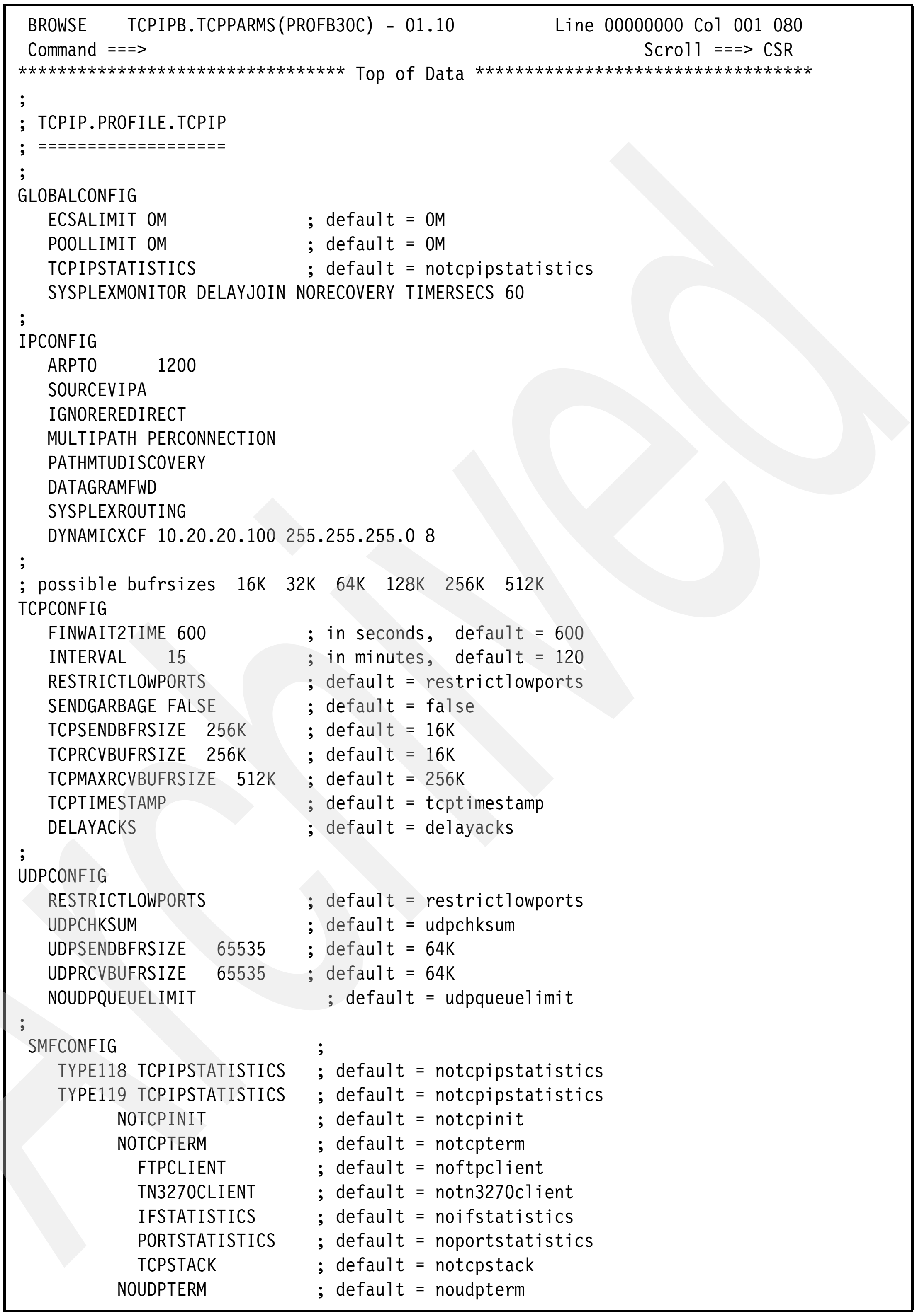
Figure E-5 SC30 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB30C
```

SC30 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD

The PROFILE for the stack is PROFB30C.



```
 BROWSE    TCPIPB.TCPPARMS(PROFB30C) - 01.10         Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************* Top of Data **********************************
;
; TCPIP.PROFILE.TCPIP
; ===================
;
GLOBALCONFIG
   ECSALIMIT 0M              ; default = 0M
   POOLLIMIT 0M              ; default = 0M
   TCPIPSTATISTICS           ; default = notcpipstatistics
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
   DATAGRAMFWD
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.100 255.255.255.0 8
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
   RESTRICTLOWPORTS          ; default = restrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   DELAYACKS                 ; default = delayacks
;
UDPCONFIG
   RESTRICTLOWPORTS          ; default = restrictlowports
   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 E-6 SC30 TCPIPB profile (PROFB30C) - continued

```
; ----------------------------------------------------------------------
 AUTOLOG 5

   OMPB

 ENDAUTOLOG
; ----------------------------------------------------------------------

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    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.20.10.241     ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server
    25 TCP SMTPB                         ; SMTP Server
    53 TCP NAMED9B                       ; Domain Name Server
    53 UDP NAMED9B                       ; Domain Name Server
   111 TCP PORTMAPB                      ; Portmap Server
   111 UDP PORTMAPB                      ; 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 10.20.10.242        ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.20.10.242        ; 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
   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             ; MVS Telnet Server SECURE
   12000 UDP NET                 ; Enterprise Extender signal data
   12001 UDP NET                 ; Enterprise Extender NETWORK priority
   12002 UDP NET                 ; Enterprise Extender HIGH priority
   12003 UDP NET                 ; Enterprise Extender MEDIUM priority
   12004 UDP NET                 ; Enterprise Extender LOW priority
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
```

Figure E-7 SC30 TCPIPB profile (PROFB30C) - continued

```
;-------------------------------------------------------------
;Hipersocks: trle definitions are dynamically created on vtam  -
;-------------------------------------------------------------
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4LNK  IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5LNK  IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6LNK  IPAQIDIO      IUTIQDF6
;
;-------------------------------------------------------------
; Static VIPA definitions                                      -
;-------------------------------------------------------------
 DEVICE STAVIPA1    VIRTUAL 0
 LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
;
;-------------------------------------------------------------
;Osa definitions... relates to sys1.vtamlst trle definitions   -
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0             -
;-------------------------------------------------------------
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 20
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0LNK  IPAQENET      OSA20A0 VLANID 21
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0LNK  IPAQENET      OSA20C0 VLANID 20
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 21
;
HOME
   10.20.4.234   IUTIQDF4LNK
   10.20.4.235   IUTIQDF5LNK
   10.20.5.236   IUTIQDF6LNK
   10.20.1.230   STAVIPA1LNK
   10.20.2.232    OSA2080LNK
   10.20.3.233    OSA20A0LNK
   10.20.2.234    OSA20C0LNK
   10.20.3.235    OSA20E0LNK
;
  PRIMARYINTERFACE STAVIPA1LNK
```

Figure E-8 SC30 TCPIPB profile (PROFB30C) - continued

```
VIPADYNAMIC
 ;-----------------------------------------------------------------
 ;  Set aside 14 addresses for use with BIND and SIOCSVIPA IOCTL    -
 ;             (10.20.10.241 thru .254)                            -
 ;-----------------------------------------------------------------
  VIPARANGE  DEFINE  255.255.255.240  10.20.10.240

 ;-----------------------------------------------------------------
 ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
 ;-----------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                       10.20.10.1    PORT 20 21
                 DESTIP 10.20.20.100
                        10.20.20.101

 ;-----------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
 ;-----------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                       10.20.10.21   PORT 23
                 DESTIP 10.20.20.100
                        10.20.20.101

 ;-----------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
 ;-----------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                       10.20.10.22   PORT 23
                 DESTIP 10.20.20.100
                        10.20.20.101

 ;-----------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
 ;-----------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payrol
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                       10.20.10.23   PORT 23
                 DESTIP 10.20.20.100
                        10.20.20.101


 ;-----------------------------------------------------------------
 ;      Distribute to 10.20.20.100 via normal XCF   (no viparoute)  -
 ;      Distribute to 10.20.20.101 via IP routing   (   viparoute)  -
 ;-----------------------------------------------------------------
;;VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
  VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa
ENDVIPADYNAMIC
```

Figure E-9 SC30 TCPIPB profile (PROFB30C) - continued

```
ITRACE OFF

START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
START OSA2080
START OSA20A0
START OSA20C0
START OSA20E0
;
******************************** Bottom of Data ********************************
```

Figure E-10 SC30 TCPIPB profile (PROFB30C) - end

SC30 OMPROUTE PROC for multiple TN3270E Telnet servers and SD

Figure E-11 shows the PROC JCL for OMPROUTE.

```
***************************** Top of Data ********************
//OMPB   PROC STDENV=STDNVB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*STDOUT   DD PATH='/etc/omproute/omproute.stdout',
//*             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
**************************** Bottom of Data *******************
```

Figure E-11 SC30 OMPROUTE PROC (OMPB)

SC30 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD

Figure E-12 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(STDNVB30) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************* Top of Data **********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB30)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB30)'
OMPROUTE_DEBUG_FILE=/etc/omproute/debug30b
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure E-12 SC30 OMPROUTE STDENV file (STDNVB30)

SC30 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD

Figure E-13 shows the config for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB30) - 01.00            Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
******************************** Top of Data **********************************
Area Area_Number=0.0.0.0
     Stub_Area=NO
     Authentication_type=None
     Import_Summaries=No;
;
OSPF RouterID=10.20.1.230;
Global_Options Ignore_Undefined_Interfaces=yes;
;
Routesa_Config Enabled=No;
;
; hipersocks  IUTIQDF4LNK
 ospf_interface ip_address=10.20.4.234
                subnet_mask=255.255.255.0
                name=IUTIQDF4LNK
                attaches_to_area=0.0.0.0
                cost0=10;
;
; hipersocks  IUTIQDF5LNK
 ospf_interface ip_address=10.20.4.235
                subnet_mask=255.255.255.0
                name=IUTIQDF5LNK
                ROUTER_PRIORITY=0
                attaches_to_area=0.0.0.0
                cost0=10;
;
; hipersocks  IUTIQDF6LNK
 ospf_interface ip_address=10.20.5.236
                subnet_mask=255.255.255.0
                name=IUTIQDF6LNK
                attaches_to_area=0.0.0.0
                cost0=10;
;
; ************************************************************
; *Static 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.
; ************************************************************
ospf_interface ip_address=10.20.1.230
           subnet_mask=255.255.255.252
           destination_addr=10.20.1.230
           name=STAVIPA1LNK
           Advertise_VIPA_Routes=HOST_ONLY
           Router_Priority=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
```

Figure E-13 SC30 OMPROUTE Config (OMPB30) - continued

```
; OSA Qdio OSA2080LNK
ospf_interface ip_address=10.20.2.232
           subnet_mask=255.255.255.0
           name=OSA2080LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20A0LNK
ospf_interface ip_address=10.20.3.233
           subnet_mask=255.255.255.0
           name=OSA20A0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20C0LNK
ospf_interface ip_address=10.20.2.234
           subnet_mask=255.255.255.0
           name=OSA20C0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
          mtu=1500;
;
; OSA Qdio OSA20E0LNK
ospf_interface ip_address=10.20.3.235
           subnet_mask=255.255.255.0
           name=OSA20E0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; ************************************************************
; *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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
```

Figure E-14 SC30 OMPROUTE Config (OMPB30) - continued

```
;
; ***********************************************************
; *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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ***********************************************************
INTERFACE
     IP_Address=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure E-15 SC30 OMPROUTE Config (OMPB30) - end

SC31 files for multiple TN3270E Telnet servers and SD scenario

The following sections are included below:

- "SC31 TN3270B Server PROC for multiple TN3270E Telnet servers and SD" on page 378
- "SC31 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD" on page 379
- "SC31 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD" on page 383
- "SC31 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD" on page 383
- "SC31 OMPROUTE PROC for multiple TN3270E Telnet servers and SD" on page 388
- "SC31 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD" on page 388
- "SC31 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD" on page 389

SC31 TN3270B Server PROC for multiple TN3270E Telnet servers and SD

Figure E-16 on page 379 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='TRC=TN',
//             PROFILE=TELNB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//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 E-16 SC31 TN3270B PROC JCL

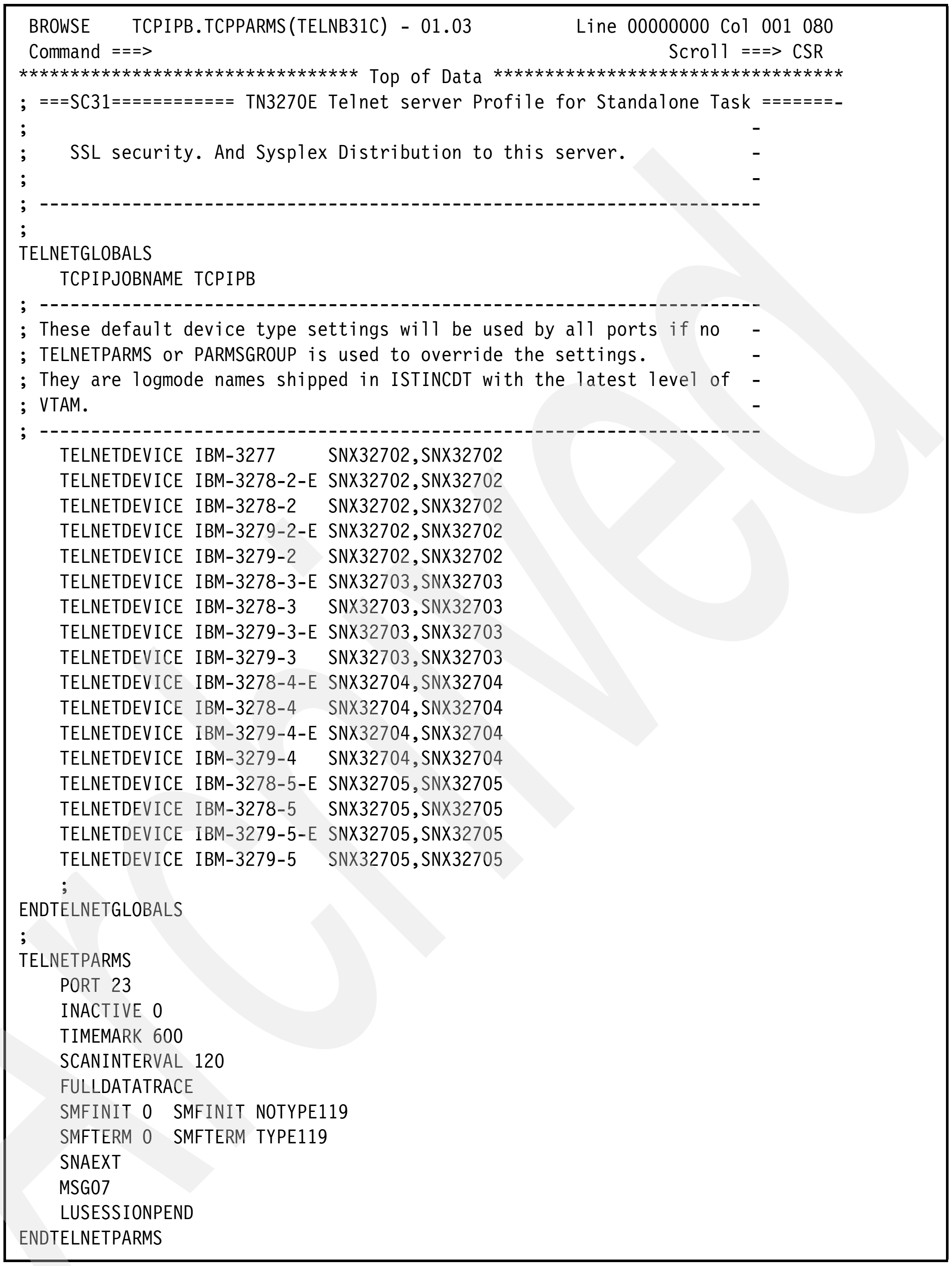
We started the server task for this scenario by issuing the following command:

```
S TN3270B,PROFILE=TELNB31C
```

SC31 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD

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).



```
 BROWSE    TCPIPB.TCPPARMS(TELNB31C) - 01.03         Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************* Top of Data **********************************
; ===SC31============ TN3270E Telnet server Profile for Standalone Task =======-
;                                                                            -
;    SSL security. And Sysplex Distribution to this server.                  -
;                                                                            -
; --------------------------------------------------------------------------
;
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 E-17   SC31 TN3270B profile (TELNB31C) - continued

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC31BB01..SC31BB99
  ENDDEFAULTLUS

  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
;
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING HFS /usr/keyring/tcpcs.kdb     ;keyring used by all secure
    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

; ---------------------------------------------------------------------
; This NOSSL        group is mapped to use no SSL security.           -
; NOLUSESSIONPEND = Terminate connection upon a logoff from the appl   -
; ---------------------------------------------------------------------
  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
```

Figure E-18 SC31 TN3270B profile (TELNB31C) - continued

```
; ----------------------------------------------------------------------
; 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

  DESTIPGROUP GENERALUSER 10.20.10.21   ENDDESTIPGROUP
  DESTIPGROUP ADMIN       10.20.10.22   ENDDESTIPGROUP
  DESTIPGROUP PAYROLL     10.20.10.23   ENDDESTIPGROUP
  DESTIPGROUP SHIPPING    10.20.1.241   ENDDESTIPGROUP
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLP0  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER      DESTIPGRP,SHIPPING
  DEFAULTAPPL TSO       DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE
 ;------------------------------------------------------------------
 ; 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 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
;
******************************** Bottom of Data ********************************
```

Figure E-19   SC31 TN3270B profile (TELNB31C) - end

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC30B*?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.
- *?* 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 decided to use 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.

SC31 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD

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("RESOLVER_CONFIG=//''TCPIPB.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=*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*TNDBCSER DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure E-20 SC31 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB31C
```

SC31 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD

The PROFILE for the stack is PROFB31C.

```
 BROWSE    TCPIPB.TCPPARMS(PROFB31C) - 01.09         Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> CSR
********************************* Top of Data **********************************
;
; TCPIP.PROFILE.TCPIP   for SC31: Sysplex distributing TN3270
; ==================
;
GLOBALCONFIG
   ECSALIMIT 0M              ; default = 0M
   POOLLIMIT 0M              ; default = 0M
   TCPIPSTATISTICS           ; default = notcpipstatistics
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
   DATAGRAMFWD
   SYSPLEXROUTING
   DYNAMICXCF 10.20.20.101 255.255.255.0 8
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
   RESTRICTLOWPORTS          ; default = restrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   DELAYACKS                 ; default = delayacks
;
UDPCONFIG
   RESTRICTLOWPORTS          ; default = restrictlowports
   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 E-21   SC31 TCPIPB profile (PROFB31C) - continued

```
; ----------------------------------------------------------------------
 AUTOLOG 5

   OMPB

 ENDAUTOLOG
; ----------------------------------------------------------------------

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    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                         ; control port
    23 TCP OMVS    BIND 10.20.10.251     ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server
    25 TCP SMTPB                        ; SMTP Server
    53 TCP NAMED9B                      ; Domain Name Server
    53 UDP NAMED9B                      ; Domain Name Server
   111 TCP PORTMAPB                     ; Portmap Server
   111 UDP PORTMAPB                     ; 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 10.20.10.252       ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.20.10.252       ; 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
   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            ; MVS Telnet Server SECURE
   12000 UDP NET                ; Enterprise Extender signal data
   12001 UDP NET                ; Enterprise Extender NETWORK priority
   12002 UDP NET                ; Enterprise Extender HIGH priority
   12003 UDP NET                ; Enterprise Extender MEDIUM priority
   12004 UDP NET                ; Enterprise Extender LOW priority
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY jOs9m2ap AGENT 161
```

Figure E-22 SC31 TCPIPB profile (PROFB31C) - continued

```
;---------------------------------------------------------------
;Hipersocks: trle definitions are dynamically created on vtam  -
;---------------------------------------------------------------
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4LNK  IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5LNK  IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6LNK  IPAQIDIO      IUTIQDF6
;
;---------------------------------------------------------------
; Static VIPA definitions                                      -
;---------------------------------------------------------------
 DEVICE STAVIPA1    VIRTUAL 0
 LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
;
;---------------------------------------------------------------
;Osa definitions... relates to sys1.vtamlst trle definitions   -
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0             -
;---------------------------------------------------------------
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 20
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0LNK  IPAQENET      OSA20A0 VLANID 21
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0LNK  IPAQENET      OSA20C0 VLANID 20
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 21
;
HOME
   10.20.4.244   IUTIQDF4LNK
   10.20.4.245   IUTIQDF5LNK
   10.20.5.246   IUTIQDF6LNK
   10.20.1.241   STAVIPA1LNK
   10.20.2.242    OSA2080LNK
   10.20.3.243    OSA20A0LNK
   10.20.2.244    OSA20C0LNK
   10.20.3.245    OSA20E0LNK
;
  PRIMARYINTERFACE STAVIPA1LNK
```

Figure E-23   SC31 TCPIPB profile (PROFB31C) - continued

```
 VIPADYNAMIC
  ;-----------------------------------------------------------------
  ;  Set aside 14 addresses for use with BIND and SIOCSVIPA IOCTL    -
  ;              (10.20.10.241 thru .254)                           -
  ;-----------------------------------------------------------------
   VIPARANGE  DEFINE  255.255.255.240  10.20.10.240

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.1   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                        10.20.10.1    PORT 20 21
                 DESTIP 10.20.20.100
                        10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.21  ;TN3270 General
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                        10.20.10.21   PORT 23
                 DESTIP 10.20.20.100
                        10.20.20.101

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.22  ;TN3270 Admin
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                        10.20.10.22   PORT 23
                 DESTIP 10.20.20.100
                        10.20.20.101

;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.20.10.23  ;TN3270 Payrol
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                        10.20.10.23   PORT 23
                 DESTIP 10.20.20.100
                        10.20.20.101


  ;-----------------------------------------------------------------
  ;      Distribute to 10.20.20.100 via IP routing   (   viparoute)  -
  ;      Distribute to 10.20.20.101 via normal XCF   (no viparoute)  -
  ;-----------------------------------------------------------------
   VIPAROUTE      DEFINE 10.20.20.100 10.20.1.230 ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.20.20.101 10.20.1.241 ; sc31's static vipa
 ENDVIPADYNAMIC
```

Figure E-24   SC31 TCPIPB profile (PROFB31C) - continued

```
ITRACE OFF

START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
START OSA2080
START OSA20A0
START OSA20C0
START OSA20E0
;
******************************** Bottom of Data ********************************
```

Figure E-25 SC31 TCPIPB profile (PROFB31C) - end

SC31 OMPROUTE PROC for multiple TN3270E Telnet servers and SD

Figure E-26 shows the PROC JCL for OMPROUTE.

```
***************************** Top of Data ********************
//OMPB   PROC STDENV=STDNVB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*STDOUT   DD PATH='/etc/omproute/omproute.stdout',
//*             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
**************************** Bottom of Data *******************
```

Figure E-26 SC31 OMPROUTE PROC (OMPB)

SC31 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD

This shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(STDNVB31) - 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=/etc/omproute/debug31b
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure E-27 SC31 OMPROUTE STDENV file (STDNVB31)

SC31 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD

The config for OMPROUTE is OMPB31.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB31) - 01.02           Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************* Top of Data **********************************
Area Area_Number=0.0.0.0
     Stub_Area=NO
     Authentication_type=None
     Import_Summaries=No;
;
OSPF RouterID=10.20.1.241;
;
Global_Options Ignore_Undefined_Interfaces=yes;
;
; AS_Boundary_routing
;  Import_Direct_Routes=yes;
;
Routesa_Config Enabled=No;
;
; hipersocks  IUTIQDF4LNK
 ospf_interface ip_address=10.20.4.244
               subnet_mask=255.255.255.0
               name=IUTIQDF4LNK
               attaches_to_area=0.0.0.0
               cost0=10;
;
; hipersocks  IUTIQDF5LNK
 ospf_interface ip_address=10.20.4.245
               subnet_mask=255.255.255.0
               name=IUTIQDF5LNK
               ROUTER_PRIORITY=0
               attaches_to_area=0.0.0.0
               cost0=10;
;
; hipersocks  IUTIQDF6LNK
 ospf_interface ip_address=10.20.5.246
               subnet_mask=255.255.255.0
               name=IUTIQDF6LNK
               attaches_to_area=0.0.0.0
               cost0=10;
;
```

Figure E-28 SC31 OMPROUTE config (OMPB31)

```
; ************************************************************
; *Static 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.
; ************************************************************
ospf_interface ip_address=10.20.1.241
           subnet_mask=255.255.255.252
           destination_addr=10.20.1.241
           name=STAVIPA1LNK
           Advertise_VIPA_Routes=HOST_ONLY
           Router_Priority=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA2080LNK
ospf_interface ip_address=10.20.2.242
           subnet_mask=255.255.255.0
           name=OSA2080LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20A0LNK
ospf_interface ip_address=10.20.3.243
           subnet_mask=255.255.255.0
           name=OSA20A0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20C0LNK
ospf_interface ip_address=10.20.2.244
           subnet_mask=255.255.255.0
           name=OSA20C0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA20E0LNK
ospf_interface ip_address=10.20.3.245
           subnet_mask=255.255.255.0
           name=OSA20E0LNK
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.0
           cost0=10
           mtu=1500;
```

Figure E-29 SC31 OMPROUTE config (OMPB31) - continued

```
;
; ************************************************************
; *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 10.20.10.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.10.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.20.10.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
; ************************************************************
; *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 10.20.20.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.20.20.* 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=10.20.20.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure E-30 SC31 OMPROUTE config (OMPB31) - end

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see "How to get IBM Redbooks" on page 395. Note that some of the documents referenced here may be available in softcopy only.

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications*, SG24-7340
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-based Network Security*, SG24-7342
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *IP Network Design Guide*, SG24-2580
- *OSA-Express Implementation Guide*, SG24-5948
- *zSeries HiperSockets*, SG24-6816
- *SNA in a Parallel Sysplex Environment*, SG24-2113
- *High Availability Considerations: SAP R/3 on DB2 for OS*, SG24-2003
- *z/OS 1.6 Security Services Update*, SG24-6448

Other publications

These 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

Online resources

These Web sites and URLs are also relevant as further information sources:

- Mainframe networking

  `http://www.ibm.com/servers/eserver/zseries/networking/`
- z/OS Communications Server product overview

  `http://www.ibm.com/software/network/commserver/zos/`
- z/OS Communications Server publications

  `http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/r8pdf/cserver.html`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications

Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications

Understand important CS for z/OS TCP/IP standard applications

See TCP/IP standard application implementation examples

Leverage TCP/IP standard applications for your needs

This new and improved *Communications Server (CS) for z/OS TCP/IP Implementation* series provides easy to understand, step-by-step guidance on enabling the most commonly used and important functions of CS for z/OS TCP/IP.

In this IBM Redbook we provide useful implementation scenarios and configuration recommendations for many of the TCP/IP standard applications supported by the z/OS Communications Server.

For more specific information about CS for z/OS base functions, high availability, and security, see the other volumes in the series. These are:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security*, SG24-7342

SG24-7340-00 ISBN 0738494577