

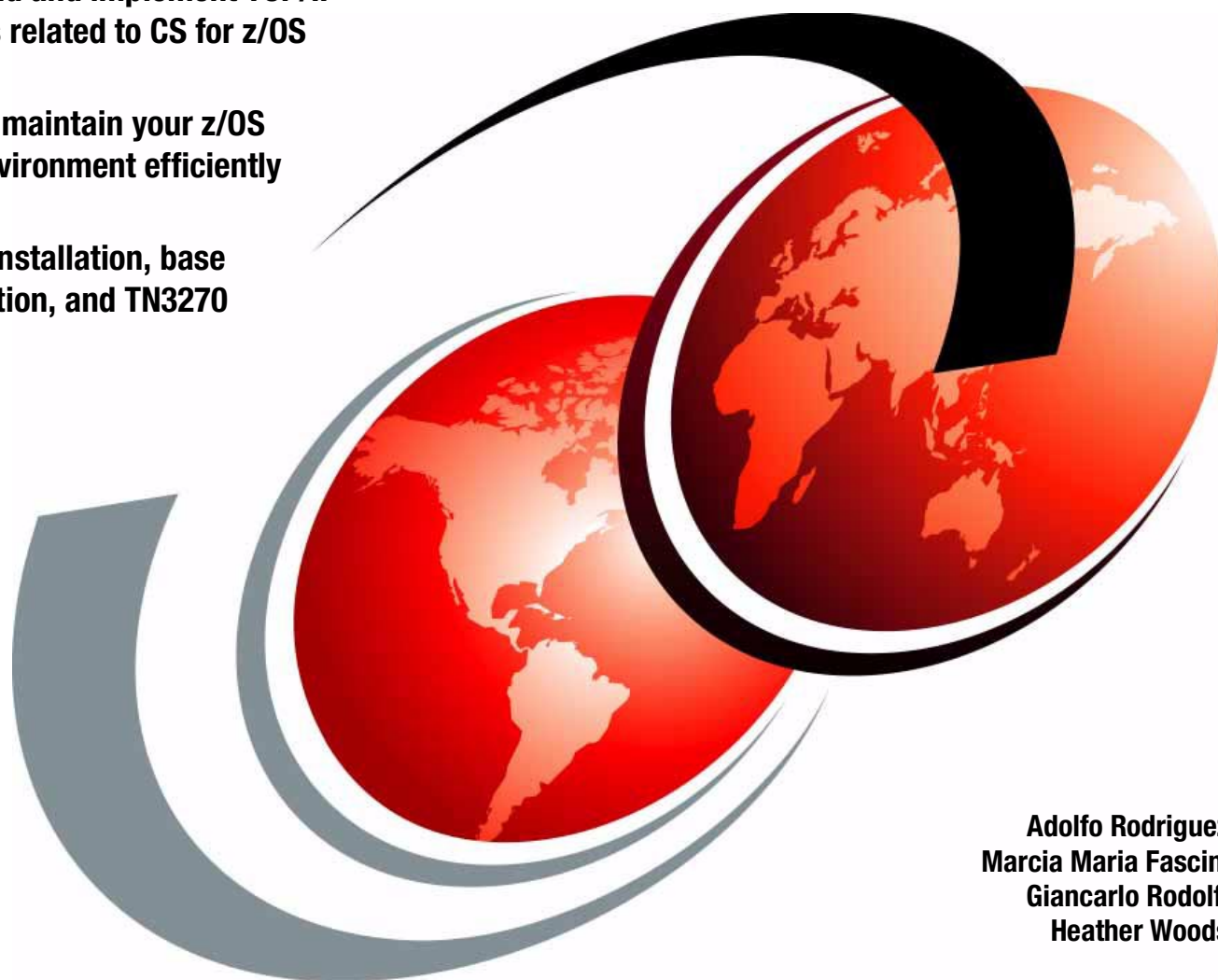
Communications Server for z/OS V1R2 TCP/IP Implementation Guide

Volume 1: Base and TN3270 Configuration

Understand and implement TCP/IP
strategies related to CS for z/OS

Build and maintain your z/OS
TCP/IP environment efficiently

Includes installation, base
configuration, and TN3270



Adolfo Rodriguez
Marcia Maria Fascini
Giancarlo Rodolfi
Heather Woods

Redbooks



International Technical Support Organization

**Communications Server for z/OS V1R2 TCP/IP
Implementation Guide Volume 1: Base and TN3270
Configuration**

June 2002

Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page ix.

Fourth Edition (June 2002)

This edition applies to Version 1, Release 2 of Communications Server for z/OS.

Note: This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

© Copyright International Business Machines Corporation 1998 2002. All rights reserved.

Note to U.S Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Contents	iii
Notices	ix
Trademarks	x
Preface	xi
The team that wrote this redbook	xii
Become a published author	xii
Comments welcome	xiii
Chapter 1. Communications Server for z/OS IP overview	1
1.1 Evolving architecture of TCP/IP on S/390	1
1.1.1 MVS OpenEdition or UNIX System Services	2
1.1.2 TCP/IP V3 for MVS and TCP/IP for MVS OpenEdition Applications Feature	3
1.1.3 OS/390 TCP/IP OpenEdition stack	6
1.1.4 OS/390 eNetwork Communications Server V2R5 IP and later	8
1.2 Functional overview of Communications Server for z/OS IP	10
1.2.1 Operating environment	10
1.2.2 Supported connectivity protocols and devices	10
1.2.3 Supported routing applications	12
1.2.4 Enterprise Extender	13
1.2.5 Application programming interfaces (APIs)	13
1.2.6 Communications Server for z/OS IP applications	14
1.2.7 Diagnostic aids	14
1.3 IBM Communications Server for z/OS V1R2 IP enhancements	14
1.3.1 Resolver changes	15
1.3.2 TN3270 Server enhancements	15
1.3.3 Usability and serviceability enhancements	15
Chapter 2. Customizing UNIX System Services	17
2.1 Customization levels of UNIX System Services	17
2.2 UNIX System Services history	18
2.3 UNIX System Services concepts	18
2.3.1 UNIX Hierarchical File System	19
2.3.2 z/OS UNIX user identification	20
2.3.3 Accessing the z/OS UNIX shells	21
2.3.4 Operating mode	22
2.3.5 UNIX System Services communication	22
2.3.6 AF_INET transport providers	25
2.4 Customization of UNIX System Services	29
2.4.1 Started task user IDs	29
2.4.2 Parmlib definitions	29
2.4.3 OMVS start-up at IPL time	32
2.4.4 OMVS displays	33
2.5 Working with UNIX System Services: interactive interfaces for the end user	35
2.5.1 Displaying OMVS processes	37
2.5.2 Working with file systems	39
2.5.3 Manipulating files and directories	41
2.5.4 Superuser mode	42

2.6	Common user errors with UNIX System Services.	44
2.6.1	Problems with the home directory.	44
2.6.2	UNIX permission bits.	45
2.6.3	Default search path and symbolic links.	46
2.6.4	Incorrect RESOLVER_CONFIG in use.	47
Chapter 3.	Installation.	49
3.1	First things first	49
3.2	Planning your installation and migration	50
3.3	Preinstallation	51
3.3.1	IP Migration Guide	52
3.3.2	z/OS Program Directory	52
3.3.3	Program support	53
3.4	Security considerations	53
3.4.1	APF authorization	53
3.4.2	RACF environment	54
3.4.3	TCP/IP server functions	57
3.4.4	TCP/IP client functions	58
3.4.5	UNIX client functions.	58
3.4.6	TCP/IP built-in security functions	58
3.5	Installation	59
3.5.1	TCP/IP configuration data set names	59
3.5.2	Installation steps	60
3.6	Message types: Where to find them	66
3.6.1	Messages with prefix of BPX	66
3.6.2	Messages with prefix of EZA.	67
3.6.3	Messages with prefix of EZB.	67
3.6.4	Messages with prefix of EZY.	67
3.6.5	Messages with prefix of EZZ.	67
3.6.6	Messages with prefix of FOM and FSUM	67
3.6.7	Eight-digit SNA sense codes and DLC codes.	67
3.7	Checklist for installation and customization	67
Chapter 4.	Configuring base functions	71
4.1	z/OS IP Configuration Wizard and msys for Setup	72
4.1.1	z/OS IP Configuration Wizard	72
4.1.2	z/OS msys for Setup	73
4.2	PROFILE.TCPIP	74
4.2.1	Displaying the TCP/IP Config	74
4.2.2	Locating PROFILE.TCPIP	76
4.2.3	Configuration features of CS for z/OS IP	76
4.2.4	System symbolics	77
4.2.5	PROFILE.TCPIP parameters	84
4.3	Configuring the system with MVS commands.	92
4.3.1	Deleting a device and adding/changing a device	92
4.3.2	Example: changing an LCS device	94
4.4	TCPIP.DATA.	98
4.4.1	Resolvers	99
4.4.2	Resolver configuration for the TCP/IP stack.	100
4.4.3	MVS application search path	101
4.4.4	z/OS UNIX application search path.	101
4.4.5	Working with TCPDATA	102
4.4.6	Testing TCPIP.DATA	105

4.5	Configuring the SITE table (HOSTS.LOCAL)	106
4.5.1	/etc/hosts	109
4.5.2	Maintaining shared source in HOSTS.LOCAL	109
4.5.3	Maintaining shared source in /etc/hosts	110
4.6	/etc/protocol	111
4.7	/etc/services	111
4.8	Starting Communications Server for z/OS IP	114
Chapter 5.	Multiple TCP/IP stacks on z/OS	117
5.1	Value of multiple concurrent copies of TCP/IP	118
5.2	Managing network attachments	118
5.2.1	Fault-tolerant network attachment	119
5.3	Performance and capacity issues: multiple stacks	120
5.4	Common Internet Physical File System (CINET PFS)	120
5.5	Port management overview	121
5.6	SMF accounting issues: multiple stacks	122
5.7	Selecting a stack	123
5.7.1	Standard servers and clients	123
5.7.2	Non-standard servers and clients	126
5.7.3	TCP/IP TSO clients	126
5.7.4	UNIX System Services clients	127
5.7.5	Selecting configuration data sets	127
5.7.6	Sharing Resolver configuration data sets between two stacks	128
5.8	Steps for installing a second stack	129
5.9	Example: implementing a two-stack configuration	130
5.9.1	Step 1: Stack name and DATASET PREFIX	130
5.9.2	Step 2: Network connections	130
5.9.3	Step 3: Alter the BPXPRMxx member	131
5.9.4	Step 4: Allocate TCPPARMS	132
5.9.5	Step 5: Create PROFILE.TCPIP	132
5.9.6	Step 6: Create TCPIP.DATA	134
5.9.7	Step 7: Create system address space JCL procedure	135
5.9.8	Step 8: Create server address space JCL procedures	136
5.9.9	Step 9: Create server-specific configuration data sets	136
5.9.10	Step 10: Update your name server	136
5.9.11	Step 11: Create REXX program to switch TSO users	137
5.9.12	Step 12: Create VTAM definitions and USS message 10 tables	137
5.9.13	Step 13: Starting the stacks	137
Chapter 6.	National Language Support (NLS)	139
6.1	Server and client translation options	140
6.2	Standard translate tables	140
6.2.1	Using your country SBCS translate table	141
6.2.2	Using your country DBCS translate table	142
6.3	Telnet use of translate tables	143
6.3.1	Telnet sessions between two z/OS or VM hosts	143
6.3.2	Telnet sessions between z/OS and other TCP/IP hosts	143
6.4	Code set conversion utilities in UNIX System Services	145
Chapter 7.	Diagnostic tools	149
7.1	DISPLAY TCPIP command	149
7.2	NETSTAT and onetstat	150
7.2.1	Routing table displays	150
7.2.2	Home addresses display	152

7.2.3	Device/link displays	152
7.2.4	Active sockets displays	153
7.2.5	Connection detail display	154
7.2.6	TCP/IP storage usage display	155
7.2.7	NETSTAT filter enhancements	155
7.2.8	NETSTAT performance counters	156
7.2.9	Monitoring Sysplex Distributor with NETSTAT	156
7.3	PING/oping and TRACERTE/otracert commands.	157
7.4	Component trace (CTRACE)	158
7.4.1	Taking a component trace	158
7.4.2	Event Trace for TCP/IP stacks (SYSTCPIP).	159
7.4.3	Sample SYSTCPIP trace	160
7.4.4	Packet trace (SYSTCPDA)	163
7.4.5	OMPROUTE trace (SYSTCPRT)	164
7.4.6	Resolver trace (SYSTCPRE)	164
7.4.7	Intrusion detection services trace (SYSTCPIS)	164
7.5	Obtaining component trace data with a dump.	164
7.6	Analyzing a trace.	165
7.6.1	Using the IPCS panels	165
7.6.2	Using IPCS and the CTRACE command	169
7.6.3	Printing a component trace.	170
7.6.4	Useful formats	171
7.7	Processing IPCS dumps	172
7.8	Configuration profile trace	172
7.9	Job log versus syslog as diagnosis tool	172
7.10	Message types: where to find them	172
Chapter 8. TN3270 Telnet server		175
8.1	Overview	176
8.1.1	Telnet functions.	178
8.1.2	Telnet printer support	181
8.2	Telnet server customization	183
8.2.1	Customizing the TCP/IP procedure.	183
8.2.2	Customizing the VTAM configuration data set	184
8.2.3	Customizing the PROFILE data set	185
8.2.4	CLID to object mapping	193
8.2.5	USS messages	207
8.2.6	Using translation tables.	212
8.3	Operating the Telnet environment.	212
8.3.1	Telnet VARY commands.	212
8.3.2	Telnet DISPLAY commands	215
8.3.3	VTAM display commands	215
8.4	Problem determination	219
8.4.1	Telnet DEBUG	219
8.4.2	Abend Trap	220
8.4.3	CTRACE	220
Appendix A. Sample REXX to create HOSTS.LOCAL from /etc/hosts		221
Related publications		223
IBM Redbooks		223
Other resources		223
Referenced Web sites		224
How to get IBM Redbooks		224

IBM Redbooks collections.....	224
Index	225

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

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

Trademarks

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

ACF/VTAM®	IBM.COM™	Redbooks Logo 
AIX®	IMS™	RISC System/6000®
AnyNet®	IPDS™	RS/6000®
APPN®	MVS™	S/390®
C/MVS™	MVS/ESA™	SecureWay®
CICS®	NetView®	SP™
DFS™	OpenEdition®	VTAM®
DFSMS/MVS®	OS/2®	z/OS™
DPI®	OS/390®	z/VM™
@server®	PAL®	zSeries™
ESCON®	Parallel Sysplex®	
FFST™	RACF®	
IBM®	Redbooks™	

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

Domino™

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

The Internet and enterprise-based networks have led to the rapidly increasing reliance upon TCP/IP implementations. The z/Series platform provides an environment upon which critical business applications flourish. The demands placed on these systems are ever-increasing and such demands require a solid, scalable, highly available, and highly performing operating system and TCP/IP component. z/OS and Communications Server for z/OS provide for such a requirement with a TCP/IP stack that is robust and rich in functionality.

The *Communications Server for z/OS TCP/IP Implementation Guide* series provides a comprehensive, in-depth survey of CS for z/OS. The series has been restructured to conform to a more task-oriented, user-friendly standard. As a result, many portions once included in Volume 1 have been moved to other volumes including security issues, network interface connectivity, and routing considerations.

In *Volume 1*, we begin by providing an introduction to CS for z/OS. We include a survey on the evolution of what was once known as TCP/IP for MVS. We cover issues involved in using UNIX System Services as well as installation and base configuration of CS for z/OS. We further discuss other stack-related issues such as language support and multi-stack environments. Finally, because the TN3270 Server is so closely integrated with the stack, this volume details the intricacies of the server.

Because of the varied scope of CS for z/OS, this volume is not intended to cover all aspects of it. The main goal is to provide sufficient detail to install and initialize the TCP/IP stack. Additionally, this volume covers all stack-related issues. That is, anything that is system- or stack-related falls into the realm of this volume. For more advanced information, including routing and network interfaces, please reference the other volumes in the series. These are:

- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- ▶ *OS/390 eNetwork Communications Server for V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing*, SG24-6516 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date July 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date July 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002))

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, Raleigh Center.

Adolfo Rodriguez is a Senior I/T Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes worldwide on all areas of TCP/IP. Before joining the ITSO, Adolfo worked in the design and development of CS for z/OS, in RTP, NC. He holds a B.A. degree in Mathematics and B.S. and M.S. degrees in Computer Science, from Duke University. He is currently pursuing the Ph.D. degree in Computer Science at Duke University, with a concentration on Networking Systems.

Marcia Maria Fascini is a Systems Specialist in Brazil. She has four years of experience in CICS and six years of experience in the networking field. She holds a degree in Mathematics from Fundação Santo André. Her areas of expertise include VTAM and TCP/IP networks.

Giancarlo Rodolfi is a zSeries FTSS for Latin America. He has 16 years of experience in the zSeries field. His areas of expertise include TCP/IP, z/VM, z/OS, UNIX System Services, CS for z/OS, security, Linux, WebSphere Application Server, firewalls, TN3270 Server, and Domino. He has written extensively on CS for z/OS TCP/IP services and security.

Heather Woods is a Network Systems Programmer with IBM Strategic Outsourcing in the UK. She has eight years of experience in S/390 systems, and has spent the past four years working mainly with TCP/IP, CS for OS/390(z/OS), NCP, and SNA.

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

Bob Haimowitz, Jeanne Tucker, Margaret Ticknor, Tamikia Barrow, Gail Christensen, Linda Robinson
International Technical Support Organization, Raleigh Center

Barry Mosakowski, Jeff Hagggar, Bebe Isrel, Van Zimmerman, Jerry Stevens, Tom Moore
Communications Server for z/OS Development, Raleigh, NC

Garth Madella
IBM South Africa

Peter Focas
IBM New Zealand

Octavio Ferreira
IBM Brazil

Steve Zammit
IBM Canada

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 team with IBM technical professionals, Business Partners and/or customers.

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 Internet note to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195



Communications Server for z/OS IP overview

Communications Server for z/OS IP provides an implementation of the TCP/IP protocol for the z/OS platform. In addition to TCP/IP, Communications Server for z/OS includes ACF/VTAM with AnyNet/MVS (also known as *Multiprotocol/HPR Services*).

This chapter provides an overview of the IP functionality included in Communications Server for z/OS and includes the following sections:

- ▶ 1.1, “Evolving architecture of TCP/IP on S/390” on page 1 describes how the TCP/IP implementation on S/390 has evolved throughout the years.
- ▶ 1.2, “Functional overview of Communications Server for z/OS IP” on page 10 provides a description of the supported functions included in the product.
- ▶ 1.3, “IBM Communications Server for z/OS V1R2 IP enhancements” on page 14 gives a summary of the new features included in the V2R10 release.

1.1 Evolving architecture of TCP/IP on S/390

Much confusion surrounds the naming of the MVS TCP/IP code that communicates with OS/390 OpenEdition MVS. Various trademark considerations have caused the names of the same (or similar) functions to fluctuate among several conventions. Some of these conventions have appeared in marketing literature and the trade press. We refer to these as *external names*. Other conventions have been used strictly as internal IBM names that the implementer may continue to hear in the marketplace or even see in system messages. We refer to these as *internal names*. Table 1-1 illustrates the many different names for CS for z/OS IP.

Table 1-1 MVS TCP/IP: Naming conventions across several releases

External Name	Internal Name
IBM TCP/IP Version 3 Release 1 for MVS	TCP/IP V3R1

External Name	Internal Name
OpenEdition MVS Applications Feature, or TCP/IP for MVS OpenEdition Applications Feature	OE Apps Feature TCP/IP OE
IBM TCP/IP Version Release 2 for MVS/ESA	TCP/IP V3R2
OpenEdition MVS Applications Feature, or TCP/IP for MVS OpenEdition Applications Feature	OE Apps Feature TCP/IP OE
eNetwork Communications Server OS/390 TCP/IP OpenEdition OS/390 TCP/IP OpenEdition for MVS/ESA and OS/390 R3/R4	Stage 1 OS/390 TCP/IP OE TCP/IP V3R3
OS/390 eNetwork Communications Server V2R5 IP	Stage 2 TCP/IP V3R4 IP V2R5 CS/390 R5
OS/390 eNetwork Communications Server V2R6 IP	There is no internal name any longer.
OS/390 eNetwork Communications Server V2R7 IP	There is no internal name any longer.
SecureWay Communications Server for OS/390 V2R8 IP	There is no internal name any longer.
IBM Communications Server for OS/390 V2R10 IP	There is no internal name any longer.
Communications Server for z/OS V1R2 IP	There is no internal name any longer.

If nothing else, the table illustrates how difficult it is to communicate with others about TCP/IP for OS/390, since the proliferation of names for TCP/IP often serves to complicate understanding.

Note: We will frequently use the shorter term *CS for z/OS IP* throughout the book to refer to *Communications Server for z/OS IP*.

In order to understand what CS for z/OS IP provides, it is helpful to look at the evolution of OpenEdition in the MVS environment and the architecture of some of the releases of TCP/IP that are illustrated in Table 1-1.

1.1.1 MVS OpenEdition or UNIX System Services

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

UNIX System Services is the OS/390 or MVS implementation of UNIX as defined by X/Open in the XPG 4.2. UNIX System Services is required for creating and using applications conforming to the POSIX or XPG4 standard. UNIX System Services coexists with traditional MVS functions and traditional MVS file types (partitioned data sets, sequential files, etc.). It concurrently allows access to HFS files and to UNIX utilities and commands by means of application programming interfaces and the interactive shell environment. MVS offers two variants of the UNIX shell environment: the OMVS shell, much like a native UNIX environment, and the lshell, an ISPF interface with access to menu-driven command interfaces.

OpenEdition programs communicate with the IP network through sockets, which are opened as AF_INET family sockets. (See “AF_INET addressing family” on page 23 for more information.) In order to interact with UNIX System Services under OS/390 and its sockets applications, an AF_INET transport provider is required. Such a transport provider can be any of the following:

- ▶ Communications Server for z/OS IP (previously known as IBM Communications Server for OS/390 IP and SecureWay Communications Server for OS/390 IP)
- ▶ AnyNet Sockets over SNA
- ▶ IPv6 demonstration stack (Web download)

Note: Although this transport is also a function IPv4 stack, it is not supported and is for demonstration purposes only.

1.1.2 TCP/IP V3 for MVS and TCP/IP for MVS OpenEdition Applications Feature

The original version of TCP/IP for MVS was ported from a VM implementation. The implementation for MVS was designed to emulate some basic VM functions for transfer of data and control information between the sockets application address spaces and the TCP/IP system address space. These functions are in a VM environment known under the names of Virtual Machine Communication Facility (VMCF) and Inter-User Communication Vehicle (IUCV). This emulation was done using the TCP/IP for MVS platform code.

The TCP/IP system address space is where the TCP/IP protocol stack is implemented in TCP/IP for MVS. The TCP/IP system address space is also often referred to as the *stack* (short for the TCP/IP protocol stack) or the *engine* (a nickname for the code that implements the TCP/IP protocol stack functions).

TCP/IP for MVS OpenEdition Applications Feature was a precursor to OS/390 TCP/IP OpenEdition and was available as a feature on both IBM TCP/IP Version 3 Release 2 for MVS and IBM TCP/IP Version 3 Release 1 for MVS. TCP/IP for MVS OpenEdition Applications Feature, however, provided only communication between IBM TCP/IP Version 3 for MVS and OS/390 OpenEdition MVS and accesses to HFS files on OpenEdition; it did not provide network connections itself. Instead, it relied on the data link control connections defined within the IBM TCP/IP Version 3 for MVS stack. In other words, TCP/IP for MVS OpenEdition Applications Feature could not run as a stand-alone TCP/IP stack, but rather had to use the services of the IBM TCP/IP Version 3 Release 1 for MVS or IBM TCP/IP Version 3 Release 2 for MVS stack. Figure 1-1 shows the relationship between IBM TCP/IP Version 3 for MVS and TCP/IP for MVS OpenEdition Applications Feature.

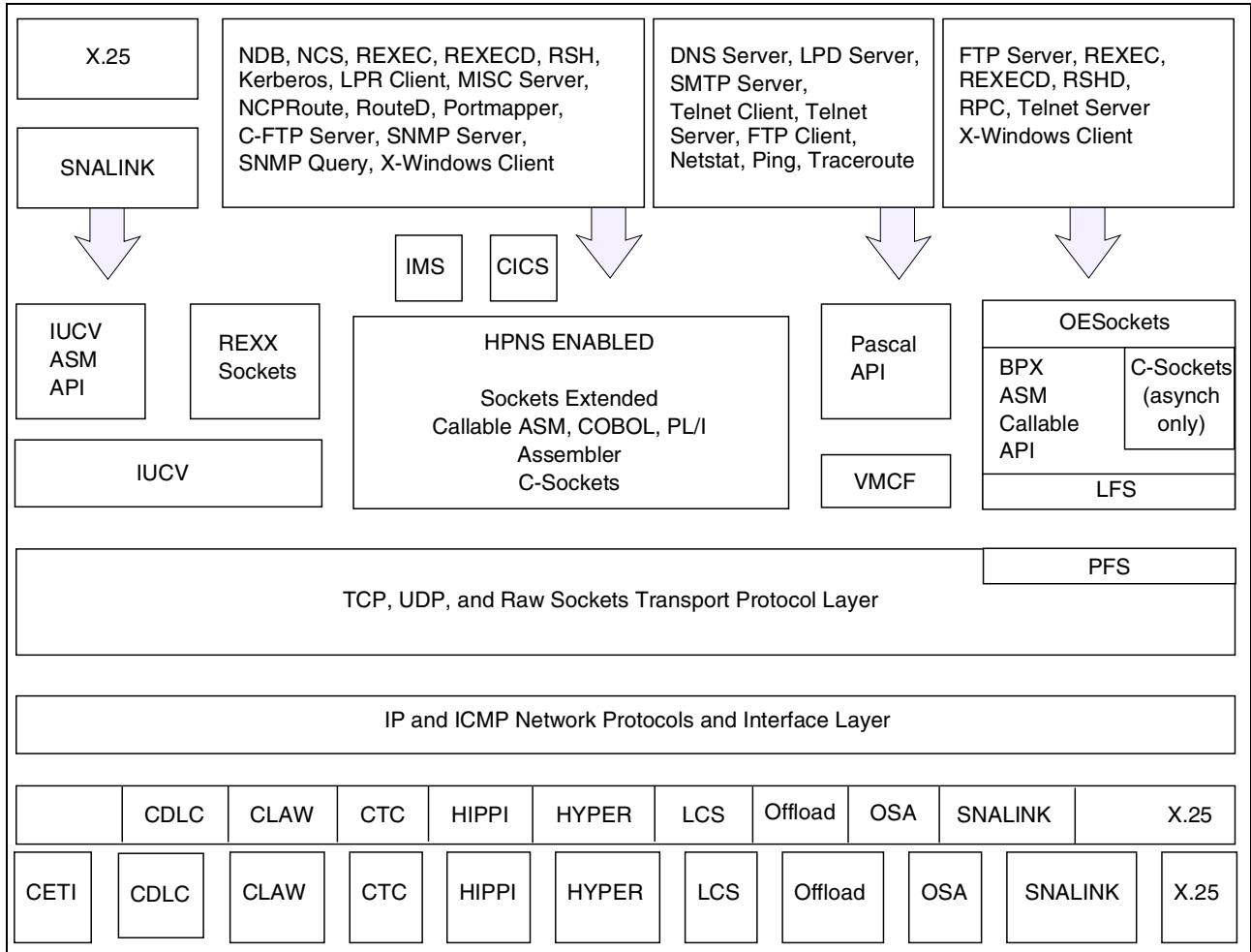


Figure 1-1 IBM TCP/IP Version 3 Release 2 for MVS with TCP/IP for MVS OpenEdition Applications Feature: Stack overview

In IBM TCP/IP Version 3 Release 2 for MVS, the TCP/IP for MVS platform was still being used, but only for the Pascal Sockets API, the IUCV Assembler Sockets API, and the REXX Sockets API, and in the rare situation where a TCP/IP C-Sockets application worked with AF_IUCV sockets (refer to Figure 1-2).

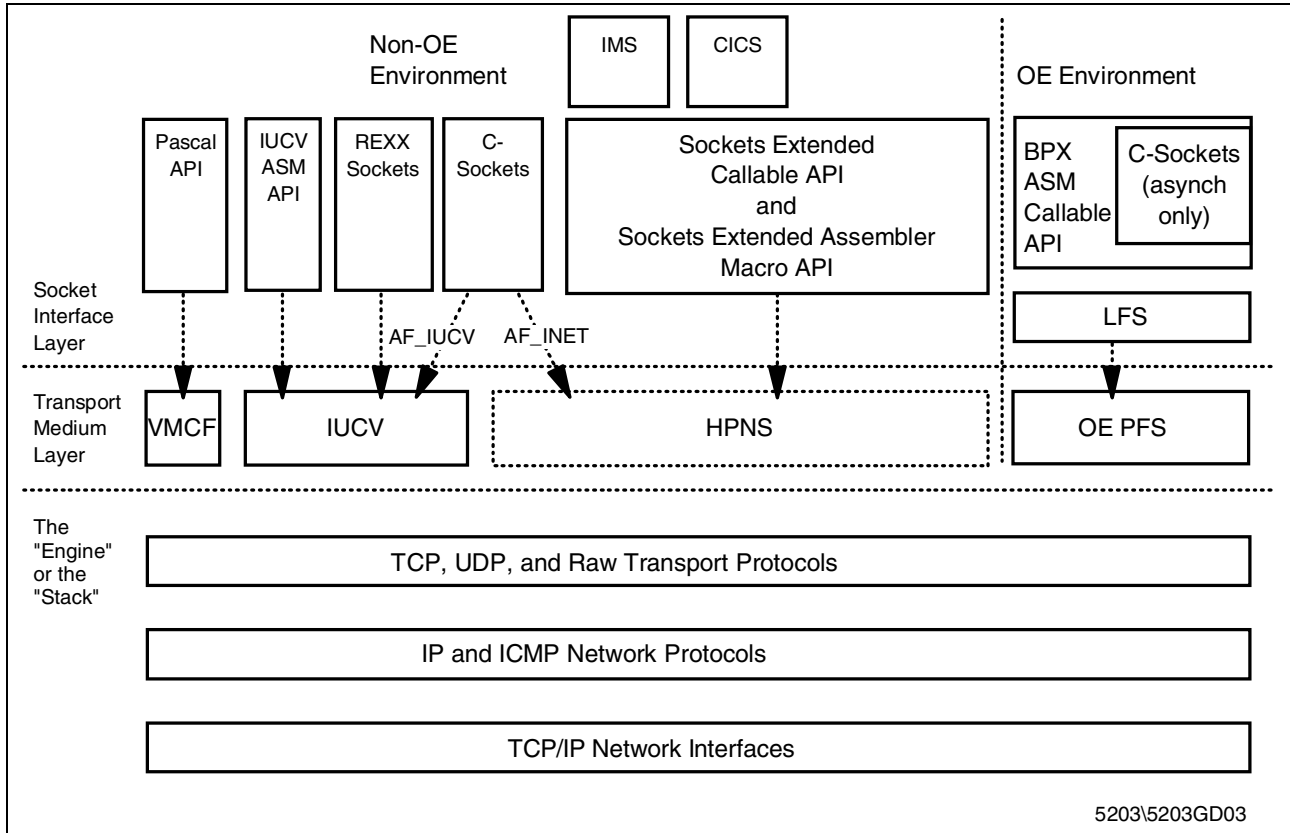


Figure 1-2 HPNS versus IUCV paths in MVS TCP/IP V3R2

As you see in Figure 1-2, the VMCF and IUCV paths both used an intermediate address space to communicate between a sockets application and the TCP/IP stack. In contrast, the High Performance Native Sockets (HPNS) path did not use any intermediate address space, thus providing a direct communication path based on OS/390 functions, such as program calls (PCs), ALETs, and data spaces. The HPNS path was an interim solution to providing performance improvements for sockets applications; this path would become superfluous with the enhanced architecture of the OS/390 V2R5 IP stack. (See more about this in 1.1.4, “OS/390 eNetwork Communications Server V2R5 IP and later” on page 8.)

Beginning with IBM TCP/IP Version 3 Release 2 for MVS, applications using the Sockets Extended interfaces (including both CICS and IMS sockets), the C-Sockets interface, and those programming interfaces that were built on top of the C-Sockets API (XTI, ONC/RPC, SNMP/DPI, NCS/RPC and X-Window) gained the full benefits of High Performance Native Sockets (HPNS). Figure 1-2 shows that there were three ways that sockets address spaces could communicate with the TCP/IP address space:

1. Via the VMCF/IUCV address space
2. Via the OpenEdition MVS Kernel address space
3. Via the High Performance Native Sockets (HPNS) path

As you saw in Figure 1-1, IBM TCP/IP Version 3 Release 2 for MVS could communicate with the OpenEdition Kernel address space using base IBM TCP/IP Version 3 for MVS, using TCP/IP for MVS OpenEdition Applications Feature, or using OS/390 TCP/IP OpenEdition (“TCP/IP V3R3”). OS/390 TCP/IP OpenEdition running in its own address space provided additional performance benefits even over IBM TCP/IP Version 3 Release 2 for MVS.

The next section presents more about the enhancements that accompanied OS/390 TCP/IP OpenEdition, which was made generally available in June 1997.

1.1.3 OS/390 TCP/IP OpenEdition stack

OS/390 TCP/IP OpenEdition was the first phase in offering native TCP/IP support in the OpenEdition environment. It was a replacement for and an enhancement to the TCP/IP for MVS Application Feature and provided full-stack TCP/IP support for the OpenEdition environment, additional connection types to the TCP/IP network, and improved performance.

OS/390 TCP/IP OpenEdition was *not* a replacement of TCP/IP for MVS V3R2 due to the fact that it did not have an equivalent level of connectivity and application support. It did, however, provide connections to OS/390 OpenEdition MVS and allowed the OpenEdition applications that ran under IBM TCP/IP Version 3 for MVS with TCP/IP for MVS OpenEdition Applications Feature to continue to run with OS/390 TCP/IP OpenEdition.

Figure 1-3 illustrates how IBM TCP/IP Version 3 Release 2 for MVS could continue to support its traditional network connections while OS/390 TCP/IP OpenEdition introduced a new connection type, MPC point-to-point, that shared the VTAM DLC code. Notice how even so-called traditional connections such as CLAW, CTC, and LCS used the common VTAM DLC code if they were defined within a separate OS/390 TCP/IP OpenEdition (TCP/IP V3R3) stack.

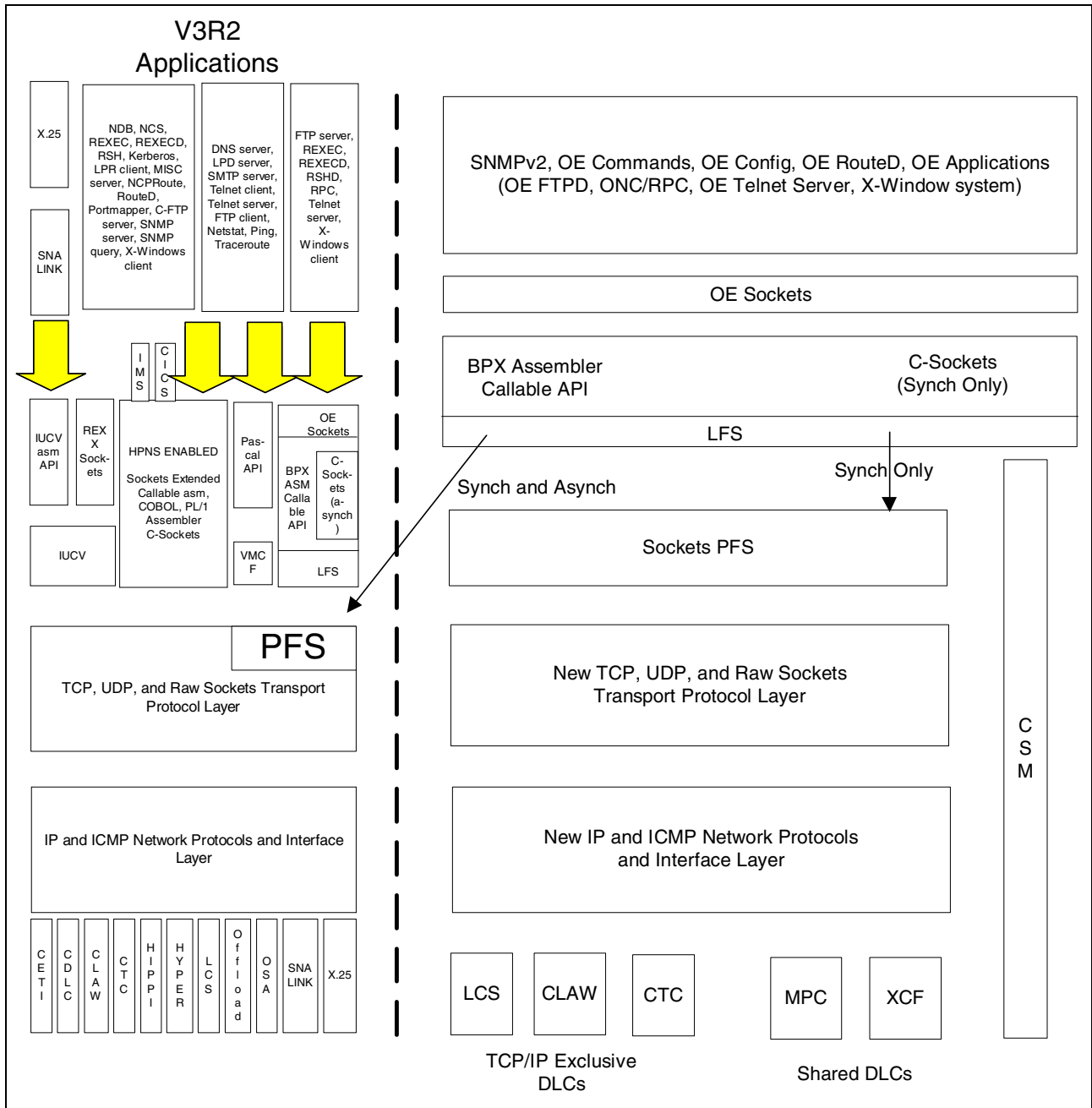


Figure 1-3 OS/390 V1R3 TCP/IP OpenEdition stack overview

As noted earlier, OS/390 TCP/IP OpenEdition could run either as a stand-alone stack or could be used in parallel with IBM TCP/IP Version 3 Release 2 for MVS. This was often necessary since the OS/390 TCP/IP OpenEdition stack did not support all the functions and network connections available with IBM TCP/IP Version 3 Release 2 for MVS. TCP/IP OS/390 TCP/IP OpenEdition and TCP/IP Version 3 Release 2 were delivered as part of the Communications Server for OS/390 Version 1 Release 3 (Program No. 5645-001) and the Communications Server for OS/390 Version 2 Release 4 (Program No. 5647-A01).

In OS/390 TCP/IP OpenEdition, the functions of the platform were replaced by a highly efficient direct communication between the OpenEdition kernel address space and a new TCP/IP stack that was integrated with OpenEdition. This communication path included the OpenEdition Physical File System (PFS) component for AF_INET (Addressing Family_Internet) sockets communication.

The OS/390 TCP/IP OpenEdition product offered many enhancements over the TCP/IP for MVS OpenEdition Applications Feature including:

- ▶ A new process model

The process model provided a fully multiprocessed environment. Also, the processing and data transfer paths were full duplex, or bi-directional. The process requests were more efficiently coded to reduce path lengths. The transport protocol layer processing exploited multiprocessing and multiprocessor environments and was completely re-entrant. All this added up to a better performing TCP/IP system.

- ▶ A new I/O process model

As you saw earlier, the TCP/IP I/O device drivers were now provided by VTAM, rather than having the I/O drivers within the TCP/IP address space. The LCS, CTC and CLAW DLCs were provided exclusively for TCP/IP. The MPC DLC could be shared between VTAM and TCP/IP. The I/O process could execute multiple I/O dispatchable units of work and was tightly integrated with the common storage management support. By providing a common I/O structure for VTAM and TCP/IP, one based on the SNA Multipath Channel (MPC) protocol, serviceability and reliability of the product improved.

- ▶ A new storage management model

OS/390 TCP/IP OpenEdition used common storage for its processing support. Therefore, buffer pool definitions did not need to be allocated in the TCP/IP address space. The storage management support handled expansion and contraction of storage resources automatically. It also handled storage requests of varying sizes and types more efficiently. The storage management support for TCP/IP was tightly integrated with the I/O model. With the use of common storage, system resources were more efficiently used.

Figure 1-3 illustrates how the Communications Storage Management (CSM) facility was used to manage communication between the Sockets PFS, through the transport protocols and network protocols, to the network interface layer of the OS/390 TCP/IP OpenEdition stack. Thus data for I/O was placed in a set of buffers from which any function all the way down the protocol stack could access it without having to move the data.

1.1.4 OS/390 eNetwork Communications Server V2R5 IP and later

OS/390 eNetwork Communications Server V2R5 IP and later was the second phase in offering native TCP/IP support in the OpenEdition environment. It was a complete replacement of and an enhancement to IBM TCP/IP Version 3 for MVS and to OS/390 TCP/IP OpenEdition. Like OS/390 TCP/IP OpenEdition, it provided full stack TCP/IP support for the OpenEdition environment, but, unlike OS/390 TCP/IP OpenEdition, it provided:

- ▶ A full array of data link control possibilities, as you see in Figure 1-4.
- ▶ Applications that were not available with OS/390 TCP/IP OpenEdition, for example, TN3270E (RFC1647).

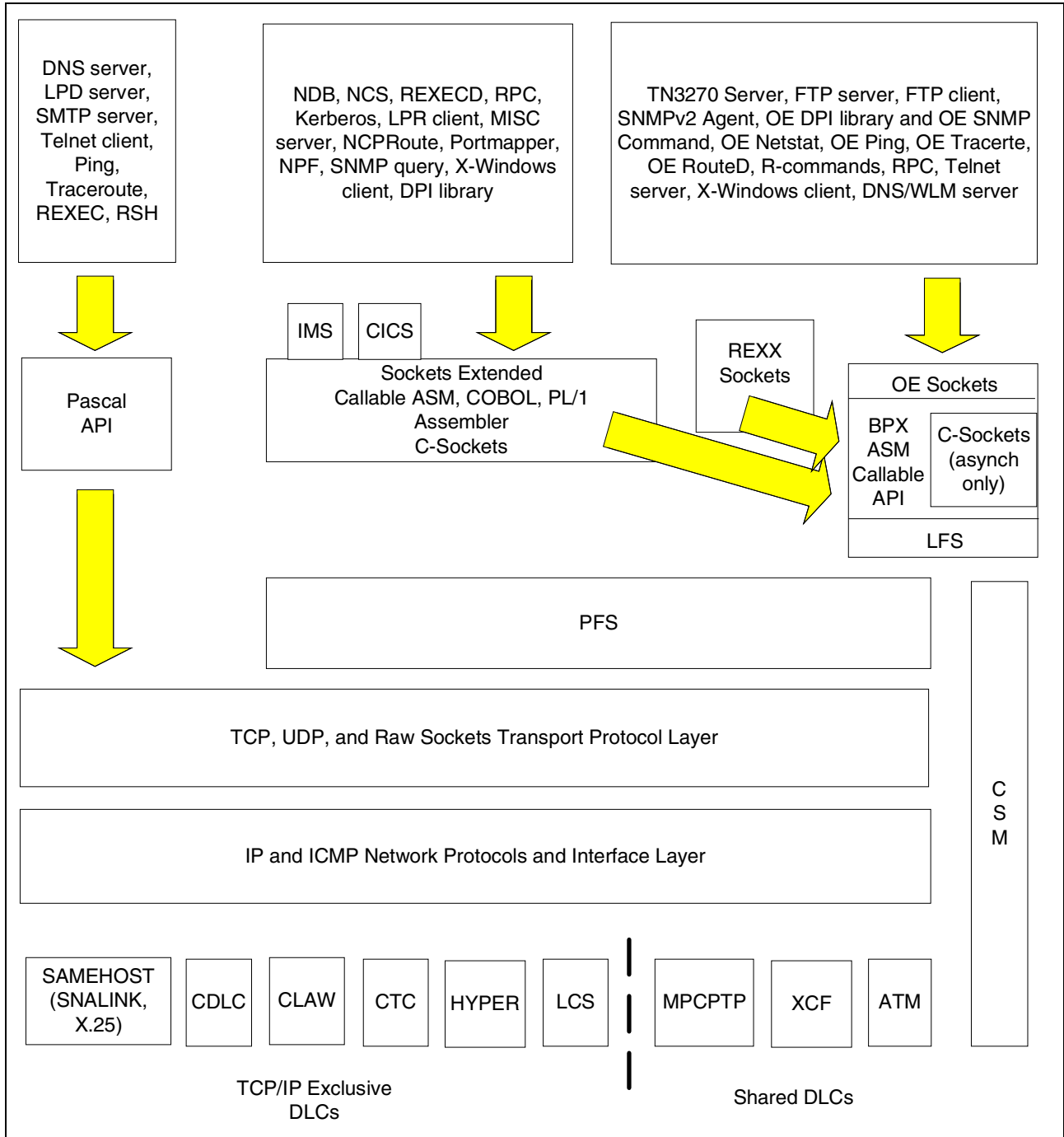


Figure 1-4 OS/390 eNetwork Communications Server V2R5 IP: Stack overview

Communications Server for z/OS IP builds upon all the enhancements introduced in OS/390 TCP/IP OpenEdition, *Stage 1*, and discussed earlier in 1.1.3, “OS/390 TCP/IP OpenEdition stack” on page 6. It expands upon what was available with OS/390 TCP/IP OpenEdition by being fully integrated with OpenEdition and exploiting UNIX System Services. In contrast to earlier TCP/IP OpenEdition interfaces, it requires some OpenEdition configuration. It runs as a single stack that serves both the traditional MVS environment and the OpenEdition environment.

The HPNS sockets introduced for performance purposes in IBM TCP/IP Version 3 Release 2 for MVS are no longer necessary in CS for z/OS IP since the latest stacks provide performance enhancements that obsolete HPNS. Any applications written to take advantage of HPNS can be seamlessly migrated to CS for z/OS IP while enjoying the full performance benefits of the fully integrated stack because they are automatically converted to OE sockets.

1.2 Functional overview of Communications Server for z/OS IP

Communications Server for z/OS IP is the second phase of the z/OS TCP/IP evolution towards using z/OS UNIX System Services. Today's CS for z/OS IP enjoys improved performance due to a redesigned stack, one that takes advantage of Communications Storage Management (CSM) and of VTAM's Multi-Path Channel (MPC) and Queued Direct I/O (QDIO) capabilities. This tight coupling with VTAM provides enhanced performance and serviceability. In CS for z/OS IP, two worlds converge, providing access to z/OS UNIX System Services and the traditional MVS environment via network attachments, both old and new.

Due to performance improvements in the stack, it is, in most cases, no longer necessary to consider running multiple stacks for performance reasons. In addition, features such as Server Bind Control make it even less necessary to run multiple stacks. Nonetheless, a multiple stack environment is still supported. One stack option of past releases, however, was removed from CS for z/OS with the withdrawal of support for High Speed Access Services in V2R10.

1.2.1 Operating environment

z/OS UNIX System Services customization is required in order to start OS/390 V2R5 IP or later successfully. This dependence on UNIX, of course, implies that z/OS administrators must be familiar with both traditional MVS commands and interfaces, as well as the newer UNIX flavors.

1.2.2 Supported connectivity protocols and devices

As seen in Figure 1-4 on page 9, DLCs can be classified into two categories: TCP exclusive DLCs and shared DLCs. TCP exclusive DLCs are those only available for the CS for z/OS IP stack and cannot be shared between multiple instances of CS for z/OS IP. The TCP exclusive DLCs supported by CS for z/OS IP include the following channel protocols:

- ▶ Channel Data Link Control (CDLC)

This protocol supports a native IP connection between CS for z/OS IP and an IP router coded within a 374x running Network Control Program (NCP) or within a 3746 9x0 channel-attached router.
- ▶ Common Link Access to Workstation (CLAW)

This protocol is used to connect the CS for z/OS IP to a 3172 running ICCP, an RS/6000, and Cisco routers supporting this interface.
- ▶ Channel-to-Channel (CTC)

This protocol is supported between two CS for z/OS IP systems and uses one read/write channel pair. Both parallel and ESCON channels are supported.
- ▶ Hyperchannel

This protocol is used to connect via the NSC A220 Hyperchannel Adapter and its descendants.

- ▶ LAN Channel Station (LCS)

This protocol is used by OSA, the 3172 running ICP, the 2216, and the 3746-9x0 MAE.

In addition, one more TCP/IP exclusive DLC protocol exists, although it does not make use of zSeries channels. Not to be confused with PTP Samehost, the SAMEHOST DLC enables communication between CS for z/OS IP and other servers running on the same MVS image. In the past, this communication was provided by IUCV. Currently, three such servers exploit the SAMEHOST DLC:

- ▶ SNALINK LU0

This server provides connectivity through SNA networks using LU0 traffic. It acts as an application to z/OS VTAM.

- ▶ SNALINK LU6.2

This server provides connectivity through SNA networks using LU6.2 traffic. It also acts as an SNA application to z/OS VTAM.

- ▶ X.25

This server provides connectivity to X.25 networks by using the NCP packet switching interface (NPSI).

Shared DLCs are those that can be simultaneously used by multiple instances of multiple protocol stacks. For example, a shared DLC may be used by one or more instances of CS for z/OS IP and one or more instances of z/OS VTAM. These shared DLCs include:

- ▶ Multipath Channel+ (MPC+)

MPC+ is an enhanced version of the Multipath Channel (MPC) protocol. It allows for the efficient use of multiple read and write channels. High Performance Data Transfer (HPDT) uses MPC+ together with Communication Storage Manager (CSM) to decrease the number of data copies required to transmit data. This type of connection can be used in two ways.

The first of these is called MPCPTP, in which CS for z/OS IP is connected to a peer IP stack in a point-to-point fashion. In this way, CS for z/OS IP can be connected to each of the following:

- Another CS for z/OS IP stack
- 2216
- RS/6000
- 3746-9x0 MAE
- Cisco routers via the Cisco Channel Interface Processor (CIP) or the Cisco Channel Port Adapter (CPA)

The second way to use MPC+ is to connect to an Open Systems Adapter (OSA). In this configuration, OSA acts as an extension of the CS for z/OS IP stack and not as a peer IP stack as in MPCPTP. The following are supported in this manner:

- OSA-2 native ATM (RFC1577)
- OSA-2 Fast Ethernet and FDDI (MPCOSA)
- OSA-Express QDIO uses MPC+ for the exchange of control signals between CS for z/OS IP and the OSA-Express

- ▶ MPCIPA (QDIO)

The OSA-Express provides a mechanism for communication called Queued Direct I/O (QDIO). Although it uses the MPC+ protocol for its control signals, the QDIO interface is quite different from channel protocols. It uses Direct Memory Access (DMA) to avoid the overhead associated with channel programs. The OSA-Express and CS for z/OS IP support Gigabit Ethernet, Fast Ethernet, Fast Token-Ring, and ATM LAN emulation.

A partnership between CS for z/OS IP and the OSA-Express Adapter provides offload of compute-intensive functions from the zSeries to the adapter. This interface is called IP Assist (IPA). Offloading reduces zSeries cycles required for network interfaces and provides an overall improvement in the OSA-Express environment compared to existing OSA-2 interfaces.

- ▶ XCF

The XCF DLC allows communication between multiple CS for z/OS IP stacks within a Parallel Sysplex via the Cross-System Coupling Facility (XCF). The XCF DLC can be defined, as with traditional DLCs, but it also supports XCF Dynamics, in which the XCF links are brought up automatically.

- ▶ PTP Samehost

Sometimes referred to as IUTSAMEH, this connection type is used to connect two or more CS for z/OS IP stacks running on the same MVS image. In addition it can be used to connect these CS for z/OS IP stacks to z/OS VTAM for the use of Enterprise Extender.

- ▶ HiperSockets

HiperSockets provides very fast TCP/IP communication between servers running in different Logical Partitions (LPARs) on a z800 or z900 CEC. The communication is through processor system memory via Direct Memory Access (DMA). The virtual servers that are so connected form a virtual LAN. HiperSockets uses internal QDIO at memory speeds to pass traffic between virtual servers.

Connectivity restrictions

Certain DLCs are no longer considered strategic for the networking environment of today and are therefore no longer supported in the OS/390 V2R5 IP and later stack:

- ▶ Offload for 3172

There are no plans to support 3172 offload in any future releases. The improved performance and the reduced CPU cycle provided by the new TCP/IP stack should substantially reduce or eliminate the benefit of offload for most environments.

- ▶ High Performance Parallel Interface (HiPPI)

- ▶ Continuously Executing Transfer Interface (CETI)

1.2.3 Supported routing applications

CS for z/OS IP ships two routing applications, ORouted and OMPROUTE. OMPROUTE and ORouted cannot run on the same TCP/IP stack concurrently. These applications add, delete, and change routing entries in the routing table and can be used as an alternative to static routes created via GATEWAY or BEGINROUTES definitions in the CS for z/OS IP profile.

ORouted

ORouted is the CS for z/OS IP implementation of the Routing Information Protocol (RIP) Version 1 (RFC 1058) and RIP Version 2 (RFC 1723). A much older application than OMPROUTE, ORouted has limitations. ORouted does not support zero subnets. In addition, ORouted does not support equal-cost multipath routes to a destination network or host. As a result, OMPROUTE is the recommended routing application (also called routing daemon).

OMPROUTE

In OS/390 V2R6 IP and later, OMPROUTE implements the Open Shortest Path First (OSPF) protocol described in RFC 1583 (OSPF Version 2) as well as RIPv1 and RIPv2. When configured properly, the OS/390 host running with OMPROUTE becomes an active OSPF and/or RIP router in a TCP/IP network. Either (or both) of these two routing protocols can be used to dynamically maintain the host routing table. Additionally, OS/390 V2R7 IP provided an OMPROUTE subagent that implements the OSPF MIB variable containing OSPF protocol and state information for SNMP. This MIB variable is defined in RFC 1850.

1.2.4 Enterprise Extender

OS/390 V2R6 and later supports Enterprise Extender (also known as HPR/IP). Enterprise Extender provides the traditional advantages of advanced peer-to-peer networking (APPN), such as class of service and transmission priority, in an IP network. With Enterprise Extender, you can configure SNA networks with the following characteristics:

- ▶ A private IP-backbone network
- ▶ Stable and reliable service for mission-critical SNA applications
- ▶ Cost-effective network provisioning

Enterprise Extender uses User Datagram Protocol (UDP) to access the IP network.

1.2.5 Application programming interfaces (APIs)

The following APIs are provided with OS/390 V2R5 IP and later:

- ▶ Pascal:
 - Various legacy applications and functions
- ▶ TCP/IP APIs:
 - IMS Sockets
 - CICS Sockets
 - C-Sockets
 - Assembler Callable Services
 - REXX Sockets
- ▶ OpenEdition APIs:
 - OpenEdition CSockets
 - OpenEdition Assembler Callable Services

Figure 1-4 on page 9 depicts the relationship of various applications to the APIs they utilize.

OpenEdition APIs span releases with OS/390 V2R5 IP and later. Any programs that the customer may have written using the IUCV or VMCF interfaces must be migrated, as detailed in *z/OS V1R2.0 CS: IP Migration*, GC31-8773.

Most applications utilizing the IUCV sockets API tend to be written in assembler. As a result, the most suitable sockets API choices for these types of applications will probably be one of the following:

- ▶ The TCP/IP Macro Sockets API (EZASMI)
- ▶ z/OS UNIX Assembler Callable Services API

Owners of VMCF/IUCV-based applications should convert them to one of the above APIs. TCP/IP Version 3 Release 2 and CS OS/390 Version 2 Release 4 are the final releases to support IUCV and VMCF application communications over the TCP/IP stack.

Any applications written to take advantage of HPNS should run unchanged in OS/390 V2R5 IP and later, since HPNS sockets are automatically converted to OE sockets. The HPNS applications will run faster in OS/390 V2R5 IP and later despite the sockets conversion routine because the stack improvements more than compensate for the additional path length of the conversion process. Some vendor applications written in Pascal are still being shipped; these will run under OS/390 V2R5 IP and later after relinking.

Since Pascal, SMSG, and SQE still use VMCF, it remains a requirement to start the VMCF and TNF subsystems for CS for z/OS IP. Technically you could run CS for z/OS IP without TNF and VMCF started, but you would have to forego the use of some application interfaces and even the TSO/E PING commands.

For more information on APIs available in Communications Server for z/OS IP, please consult *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228 and *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229.

1.2.6 Communications Server for z/OS IP applications

CS for z/OS IP ships with a number of client and server applications, most of which are discussed in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228 and *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229. Some application functions are relevant to this volume, however.

The TN3270 server shipped with CS for z/OS IP runs in the same address space as the stack itself. In this regard, this server application can be viewed as an extension to the stack. As a result, the configuration of this server is included in this volume and covered in Chapter 8, “TN3270 Telnet server” on page 175.

1.2.7 Diagnostic aids

CTRACE for diagnosis purposes was originally introduced for TCP/IP with IBM TCP/IP Version 3 Release 2 for MVS. With OS/390 V2R5 IP and later, all stack components use CTRACE; even PKTTRACE employs CTRACE instead of GTF. ISPF panels and REXX CLISTS are made available to the implementer to aid in IPCS dump processing.

1.3 IBM Communications Server for z/OS V1R2 IP enhancements

In this section, we outline the enhancements included with IBM Communications Server for z/OS V1R2 IP relevant to this volume. For enhancements in all other areas, please consult the other V1R2 versions of each volume in the series:

- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing*, SG24-6516 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date July 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date July 2002))

- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002))

1.3.1 Resolver changes

A new consolidated Resolver provides consistency and enhanced functionality. Various applications previously provided their own Resolver libraries to handle socket API calls such as `gethostbyname` and `gethostbyaddr`. These APIs have been enhanced to utilize a common Resolver component that allows for consistent results in Resolver calls regardless of the API being used.

1.3.2 TN3270 Server enhancements

The TN3270E Server includes several enhancements, including support for the latest TN3270E standards and enhanced security features. These are:

- ▶ Ability to dynamically update the certificate key ring file used for TN3270E SSL connections without requiring a recycle of the server.
- ▶ Contention Resolution Enhancements. The current TN3270E standards specification (RFC2355) contains certain limitations that stem from the fact that SNA is a send/receive state oriented protocol, while TN3270E is a relatively state-free protocol.
- ▶ SNA Sense Support - When the server and client operate in an SNA environment, it is impractical to perpetuate the one-byte error code mapping style of the TN3270E protocol. Especially, when SNA already provides a table of defined Sense codes. The SNA Sense Code function allows the client to return SNA Sense codes to the server, which are in turn forwarded to the SNA host as a negative response.
- ▶ Enhanced LU mapping support for dynamic IP environments. This enhancement further extends an installation's ability to define rules that assign LUs for clients that are assigned dynamic IP addresses. Specifically, installations with TN3270E clients using dynamic IP that are establishing TN3270E SSL protected sessions can now assign LUs based on the user ID that corresponds to the client's digital certificate. When this feature is enabled, the TN3270E server queries RACF with the client's certificate and obtains the user ID associated with the certificate. This user ID can then be used in TN3270E server LU assignment policy to assign a desired LU.

1.3.3 Usability and serviceability enhancements

Communications Server for z/OS V1R2 provides a number of usability and serviceability enhancements including:

- ▶ `msys` support

The z/OS Managed System Infrastructure (`msys`) is a z/OS component that simplifies system management process for z/OS elements. It provides the system administrator with a consistent graphical user interface that can be used to customize and configure z/OS elements. In this release, Communication Server provides `msys` support that allows the network administrator to configure the TCP/IP protocol stack using the `msys` GUI.
- ▶ Improved storage monitoring

Enhancements to the storage monitoring and management facilities for TCP/IP allow for more effective monitoring and management of storage used by the Communication Server. These enhancements allow users to monitor TCP/IP storage usage with a new operator command.

- ▶ TCP/IP SMF recording enhancements

All TCP/IP SMF records now follow a new standard record format. The new format allows TCP/IP to provide a significant amount of new data in several new SMF records that can be used for an installation's capacity planning, tuning and/or accounting procedures.



Customizing UNIX System Services

Communications Server for z/OS IP requires that UNIX System Services be customized in *full-function mode* before the TCP/IP stack will successfully initialize.

Note: In this redbook we use the terms UNIX System Services, OMVS, and OS/390 OpenEdition interchangeably. With OS/390 V2R6, the name OpenEdition was changed to UNIX System Services.

This chapter is designed to give you an overview of UNIX System Services, an appreciation for the coding and security considerations involved with UNIX System Services, and insight into some common user errors with UNIX System Services.

This chapter contains the following sections:

- ▶ 2.1, “Customization levels of UNIX System Services” on page 17
- ▶ 2.2, “UNIX System Services history” on page 18
- ▶ 2.3, “UNIX System Services concepts” on page 18
- ▶ 2.4, “Customization of UNIX System Services” on page 29
- ▶ 2.5, “Working with UNIX System Services: interactive interfaces for the end user” on page 35
- ▶ 2.6, “Common user errors with UNIX System Services” on page 44

2.1 Customization levels of UNIX System Services

There are two levels of z/OS UNIX services:

- ▶ *Minimum mode*, indicating that although OMVS initializes, it provides few z/OS UNIX services, and there is no support for TCP/IP and the z/OS shell. In this mode there is no need for DFSMS or for a security product such as RACF.
- ▶ *Full-function mode*, indicating that the complete array of z/OS UNIX services is available. In this mode DFSMS, RACF, and the Hierarchical File System (HFS) are required. TCP/IP

and HFS interaction with UNIX System Services is defined within the BPXPRMxx member of SYS1.PARMLIB.

z/OS V1R2.0 UNIX System Services Planning, SA22-7800 provides a good description of the UNIX System Services customization process. It also includes a chapter devoted to TCP/IP.

2.2 UNIX System Services history

Beginning with MVS/ESA Version 4.3, a new type of application program interface was added to the MVS platform with the intent of integrating the UNIX operating system into MVS. Both a C programming API and an interactive environment called the *shell* were defined to interoperate with UNIX-style files, called the Hierarchical File System (HFS). Over time, other organizations developed approaches to working with UNIX on various platforms until finally an organization named X/Open documented standards of what to implement for UNIX interfaces in a series of guides published as the *X/Open Portability Guides* (XPG). X/Open now owns the term UNIX and certifies different implementations of UNIX according to the UNIX definitions contained in XPG 4.2. In 1996, the IBM OS/390 OpenEdition was awarded UNIX 95 brand certification, thus confirming that it is compliant with all current open-industry standards. In 1998 IBM changed the name from OS/390 OpenEdition to OS/390 UNIX System Services.

UNIX System Services is the z/OS implementation of UNIX as defined by X/Open in the XPG 4.2. UNIX System Services coexists with traditional MVS functions and traditional MVS file types (partitioned data sets, sequential files, etc.). It concurrently allows access to HFS files and to UNIX utilities and commands by means of application programming interfaces and the interactive shell environment. z/OS offers two variants of the UNIX shell environment: the z/OS shell (the default shell) and the tcsh shell (an enhanced version of the Berkeley UNIX C shell).

2.3 UNIX System Services concepts

z/OS UNIX enables two open systems interfaces on the z/OS operating system: an application program interface (API) and an interactive shell interface.

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

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

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

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

Prior to OS/390 V2R5, UNIX System Services required APPC/MVS in order to provide address spaces when programs issued the `fork()` or `spawn()` function of OpenEdition callable services. APPC/MVS is no longer required for this purpose in OS/390 V2R5 or later; in z/OS UNIX, forked and spawned address spaces are provided by Workload Manager (WLM).

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

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

The types of processes can be:

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

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

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

2.3.1 UNIX Hierarchical File System

Data sets and files are comparable terms. If you are familiar with MVS, you probably use the term “data set” to describe a unit of data storage. If you are familiar with AIX or UNIX, you probably use the term “file” to describe a named set of records stored or processed as a unit. In the UNIX System Services environment, the files are arranged in a Hierarchical File System (HFS).

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

- ▶ Directories, which contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside-down tree, with the root directory at the top and branches at the bottom.
- ▶ HFS files, which contain data or programs. A file containing a load module, shell script, or REXX program is called an executable file. Files are kept in directories.
- ▶ Additional local or remote file systems, which are mounted on directories of the root file system or of additional file systems.

To the MVS system, the UNIX file hierarchy appears as a collection of HFS-type data sets. Each HFS data set is a mountable file system. The root file system is the first file system mounted. Subsequent file systems can be logically mounted on a directory within the root file system or on a directory within any mounted file system.

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

For more information about HFS, please refer to the *z/OS V1R2.0 CS: IP Migration*, GC31-8773 and *z/OS V1R2.0 UNIX System Services Planning*, SA22-7800.

HFS definitions in BPXPRMxx

To get UNIX System Services active in full-function mode, you need the root file system defined in the BPXPRMxx member of SYS1.PARMLIB. The root file system is usually loaded or copied at z/OS installation time. The BPXPRMxx definition is as follows:

```
ROOT    FILESYSTEM('OMVS.SA03.ROOT')
        TYPE(HFS)
        MODE(RDWR)
```

Figure 2-1 Root HFS in BPXPRMxx

An important part of your HFS is located in the /etc directory. The /etc directory contains some basic configuration files of UNIX System Services and most applications keep their configuration files in there as well. To avoid losing all of your configuration when you upgrade your operating system, it is recommended that you put the /etc directory in a separate HFS data set and mount it at the /etc mountpoint.

```
MOUNT  FILESYSTEM('OMVS.SA03.ETC')
        MOUNTPOINT('/etc')
        TYPE(HFS)
        MODE(RDWR)
```

Figure 2-2 ETC HFS in BPXPRMxx

Note: Before you mount the new file system at the /etc mountpoint permanently, you should mount the new HFS temporarily at a different mountpoint and copy the contents of the /etc directory to the new HFS.

2.3.2 z/OS UNIX user identification

All users of an MVS system, including users of z/OS UNIX functions, must have a valid MVS user ID and password. To use standard MVS functions, the user must have the standard MVS identity based on the RACF user ID and group name.

If a unit of work in MVS uses z/OS UNIX functions, this unit of work must, in addition to a valid MVS identity, have a z/OS UNIX identity. A z/OS UNIX identity is based on a UNIX user ID (UID) and a UNIX group ID (GID). Both UID and GID are numeric values ranging from 0 to 2147483647 ($2^{31}-1$). In a z/OS UNIX system, the UID is defined in the OMVS segment in the user's RACF user profile, and the GID is defined in an OMVS segment in the group's RACF group profile. What we in an MVS environment call the user ID is in a UNIX environment normally termed the user name or the login name. It is the name the user uses to present himself or herself to the operating system. In both a z/OS UNIX system and other UNIX systems, this user name is correlated to a numeric user identification, the UID, which is used to represent this user wherever such information has to be stored in the z/OS UNIX environment. One example of this is in the Hierarchical File System, where the UID of the owning user is stored in the file security portion of each individual file.

Access to resources in the traditional MVS environment is based on the MVS user ID, group ID, and individual resource profiles that are stored in the RACF database.

Access to z/OS UNIX resources is granted *only* if the MVS user ID has a valid OMVS segment with an OMVS UID or if a default user is configured as explained below. Access to resources in the Hierarchical File System is based on the UID, the GID, and file access permission bits that are stored with each file. The permission bits are three groups of three bits each. The groups describe:

- ▶ The owner of the file itself
- ▶ The users with the same GID as the owner
- ▶ The rest of the world

The three bits are:

- ▶ Read access
- ▶ Write access
- ▶ Search access if it is a directory or if it is a file that is executable

The superuser UID has a special meaning in all UNIX environments, including the z/OS UNIX environment. This user has a UID of zero and can access every resource.

In lieu of or in addition to RACF definitions for individual users, you may define a *default user*. The default user will be used to allow users without an OMVS segment defined to access UNIX System Services. The default user concept should be used judiciously since it could become a security exposure.

You will also find more information on the RACF security aspects of implementing Communications Server for z/OS IP in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)), *z/OS V1R2.0 CS: IP Migration*, GC31-8773, and *z/OS V1R2.0 UNIX System Services Planning*, SA22-7800.

2.3.3 Accessing the z/OS UNIX shells

The following ways are available to access the z/OS UNIX shells:

- ▶ The TSO/E **OMVS** command provides a 3270 interface.
- ▶ The TSO/E **ISHELL** command provides a 3270 interface that uses ISPF dialogs.
- ▶ The **rlogin** command provides an ASCII interface.
- ▶ The **Te1net** command provides an ASCII interface.
- ▶ RS/6000 serial attached terminals, using Communications Server support.

Communications Server is a terminal attachment capability for OpenEdition. Communications Server consists of an RS/6000 that is LAN- or channel-attached to the OpenEdition host system. Using this connection, terminals on asynchronous ports on the RS/6000 can operate as if they were directly attached to the OpenEdition system.

- ▶ From a TCP/IP network, the TN3270(E) command, which provides a full-screen 3270 interface for executing the OMVS or ISHELL commands.

There are two shells, the z/OS shell and the tcsh shell. The login shell is determined by the PROGRAM parameter in the RACF OMVS segment for each user. The default is the z/OS shell.

Further information on the z/OS UNIX shells can be found in *z/OS V1R2.0 UNIX System Services User's Guide*, SA22-7801.

2.3.4 Operating mode

When a user first logs on to the z/OS UNIX shell, the user is operating in line mode. Depending on the method of accessing the shell, the user may then be able to use utilities that require raw mode (such as vi) or run an X-Windows application.

The different workstation operating modes are:

- ▶ Line mode
Input is processed after you press Enter. This is also called canonical mode.
- ▶ Raw mode
Each character is processed as it is typed. This is also called non-canonical mode.
- ▶ Graphical mode
This is a graphical user interface for X-Windows applications.

2.3.5 UNIX System Services communication

A socket is the endpoint of a communication path; it identifies the address of a specific process at a specific computer using a specific transport protocol. The exact syntax of a socket address depends on the protocol being used, that is, on its *addressing family*. When you obtain a socket via the `socket()` system call, you pass a parameter that tells the socket library to which addressing family the socket should belong. All socket addresses within one addressing family use the same syntax to identify sockets.

Socket addressing families in UNIX System Services

In a z/OS UNIX environment, the most widely used addressing families are `AF_INET` and `AF_UNIX`. There is some IPv6 support (and hence the `AF_INET6` addressing family) in Communications Server for z/OS IP in a single transport driver environment. Socket applications written to the IPv6 APIs can use the z/OS TCP/IP stack, but there is no support for IPv6 network connectivity. For this reason, the following discussion will concentrate on the `AF_UNIX` and `AF_INET` addressing families.

z/OS UNIX implements support for a given addressing family through different physical file systems. There is one physical file system for the `AF_INET` addressing family and there is another for the `AF_UNIX` addressing family. A PFS is the part of the z/OS UNIX operating system that handles the storage of data and its manipulation on a storage medium. (For more on this subject see *z/OS V1R2.0 UNIX System Services Planning*, SA22-7800.) You must know which addressing family you are using in order to code correctly in the UNIX System Services environment.

AF_UNIX addressing family

If two socket applications on the same MVS image want to communicate with each other, they may open a socket as an `AF_UNIX` family socket. In that case, the z/OS UNIX kernel address space will handle the full communication between the two applications (Figure 2-3). That is, the `AF_UNIX` physical file system is self-contained within z/OS UNIX and does not rely on other products to implement the required functions.

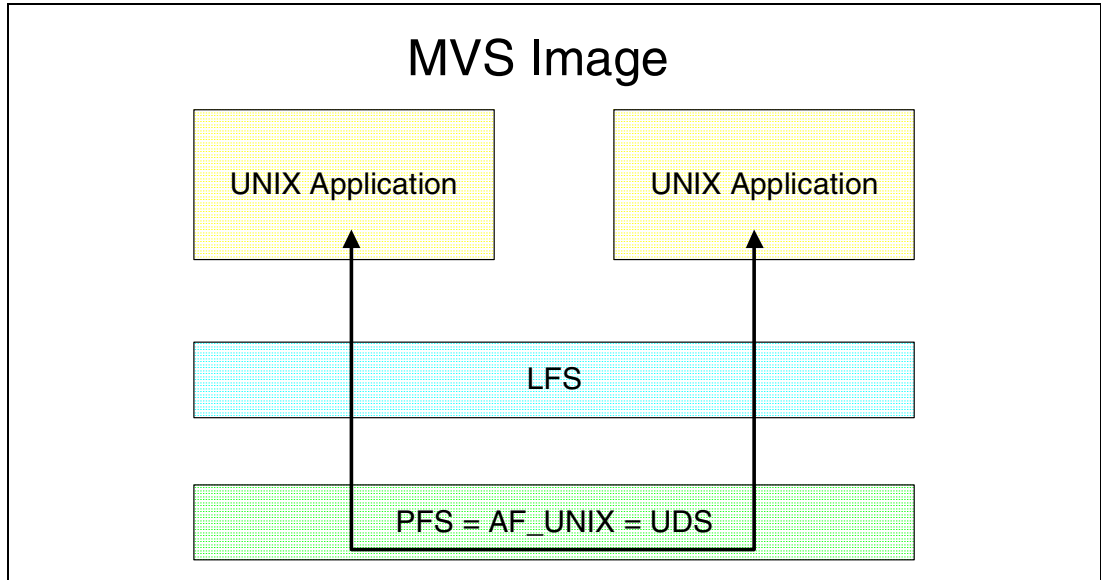


Figure 2-3 AF_UNIX sockets

AF_INET addressing family

Socket programs communicate with socket programs on other hosts in the IP network using AF_INET family sockets which, in turn, use the AF_INET physical file system.

The AF_INET physical file system relies on other products to provide the AF_INET transport services to interact with UNIX System Services and its sockets programs. For production environments, such a transport provider can be either of the following:

- ▶ Communications Server for z/OS IP
- ▶ IBM Communications Server for z/OS AnyNet

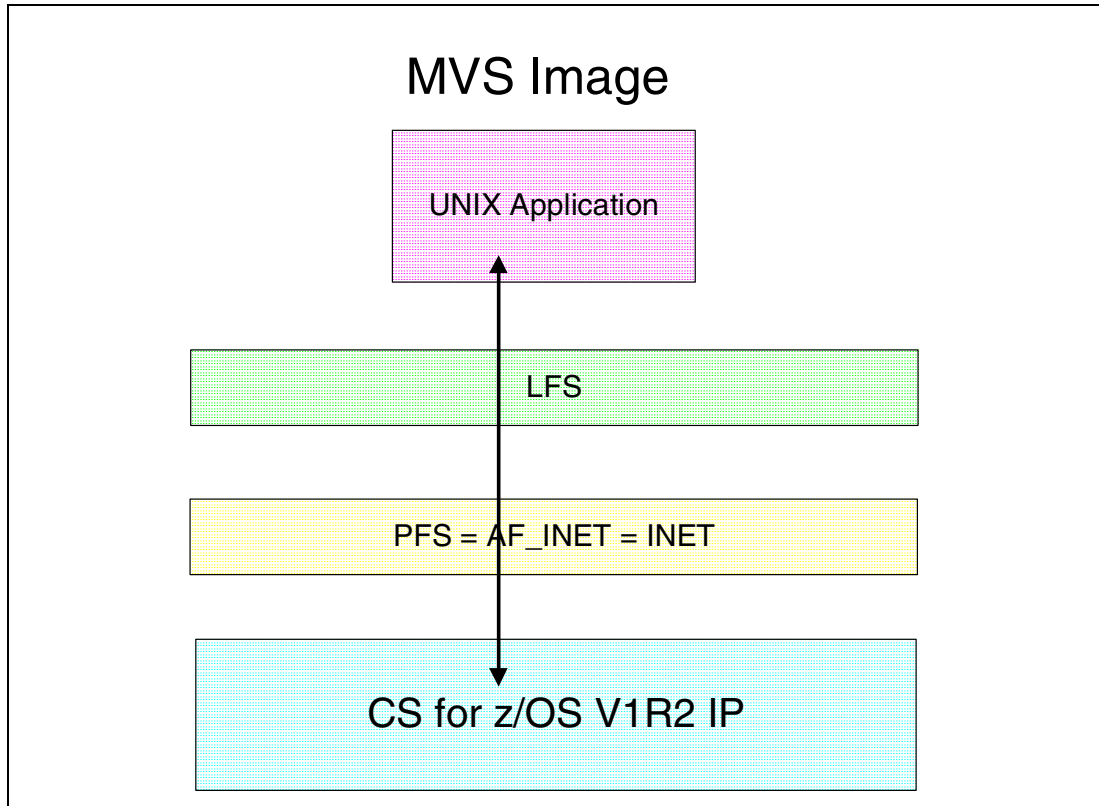


Figure 2-4 AF_INET sockets

For AF_INET sockets, the z/OS UNIX kernel address space routes the socket request to the TCP/IP system address space directly. As you see in Figure 2-4, the sockets/Physical File System layer is a transform layer between z/OS UNIX and the TCP/IP stack.

The sockets/PFS effectively transforms the sockets calls from the z/OS UNIX interface to the TCP/IP stack regardless of the version of MVS or TCP/IP. The sockets/PFS handles the communication between the TCP/IP system address space and the z/OS UNIX kernel address space in much the same manner as HPNS handles the communication between the TCP/IP system address space and the TCP/IP client and server address spaces starting with the IBM TCP/IP Version 3 Release 2 for MVS environment.

Address syntax: AF_INET and AF_UNIX

The following is a detailed description of the addressing syntax of the two addressing families, AF_INET and AF_UNIX:

► AF_INET T

The Internet addressing family, also referred to as the Internet domain.

This addressing family is used within the TCP/IP domain to identify sockets on IP hosts. A socket address in AF_INET consists of the following:

Family	1 byte binary with a value of 2, which identifies the socket address as belonging to the AF_INET addressing family.
Port	Half-word binary with port number that identifies the process.
IP address	Full-word binary with IP address of IP host in network byte order format.
Reserved	8 reserved bytes.

The following is an example of an AF_INET address that represents the Telnet server (port number 23) on an IP host with the IP address of 9.24.104.126:

```
{AF_INET 23 9.24.104.126}
```

► **AF_UNIX**

The UNIX addressing family, also referred to as the UNIX domain.

You can use AF_UNIX with UNIX sockets, where this addressing family is used for interprocess communication between UNIX processes within one MVS operating system. The syntax of an AF_UNIX address is as follows:

Family	1 byte binary with a value of 1, which identifies the socket address as belonging to the AF_UNIX addressing family.
Path	108 characters defining a path name (similar to a Hierarchical File System path name) by which this local process wants to be known by other local processes.

The following is an example of an address in the AF_UNIX addressing family:

```
AF_UNIX /u/xyz/testsrv
```

2.3.6 AF_INET transport providers

TCP/IP requires the use of the AF_INET Physical File System. The AF_INET PFS can be configured in two ways: the Integrated Sockets File System type (INET) or the Common INET Physical File System type (CINET). INET is used in a single-stack environment and CINET is used in a multiple-stack environment.

Whether to use a single AF_INET transport provider

If your background is in a UNIX environment, this may seem to be a strange question to ask, since you are used to the TCP/IP protocol stack being an integral part of the UNIX operating system. This is not the case in a z/OS environment. In this environment you may start multiple instances of a TCP/IP protocol stack, each stack running on the same operating system, but each stack having a unique TCP/IP identity in terms of network interfaces, IP addresses, host name, and sockets applications.

A simple example of a situation where you have more TCP/IP stacks running in your z/OS system is if you have two separate IP networks, one production and one test (or one secure and one not); you do not want routing between them, but you want to give hosts on both IP networks access to your z/OS environment. In this situation you could implement two TCP/IP stacks, one connected to the production IP network and another connected to the test network.

This multi-stack implementation in which you share the UNIX System Services across multiple TCP/IP stacks provides challenges. Sockets applications that need to have an affinity to a particular stack need special considerations, in some cases including the coordination of port number assignments in order to avoid conflicts. This subject is handled in depth in Chapter 5, “Multiple TCP/IP stacks on z/OS” on page 117 and in individual chapters in which port number assignments become important.

If a single AF_INET transport provider is sufficient, use the Integrated Sockets physical file system (INET). If you need more than one AF_INET transport provider, you must use the Common INET physical file system (CINET).

You can customize z/OS to use the Common INET physical file system with just a single AF_INET transport provider, but it is generally not recommended due to a slight performance decrease as compared to the Integrated Sockets Physical File System (INET). However, you may consider doing this if you expect to run multiple stacks in the future.

If you have a single AF_INET transport provider on an MVS image, such as a single TCP/IP stack or a single AnyNet stack, you probably should use the INET, as you see in Figure 2-5.

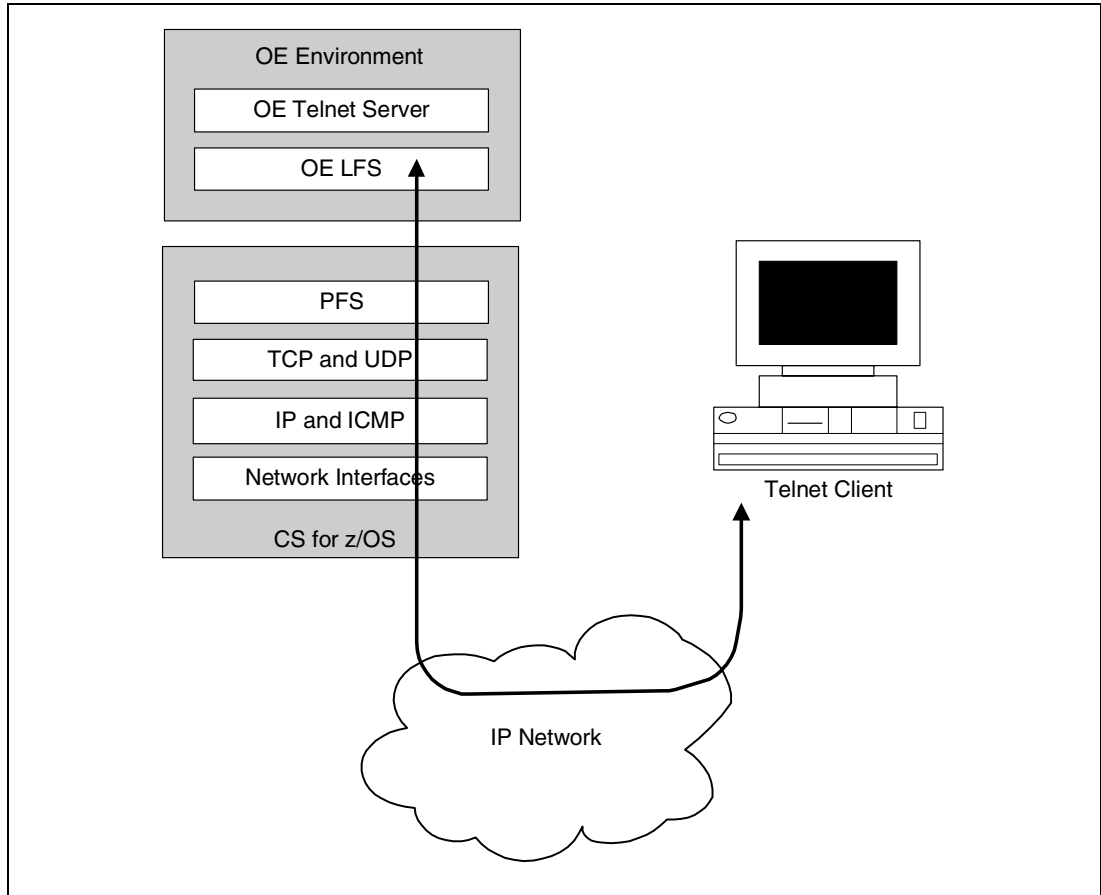


Figure 2-5 AF_INET sockets and Physical File System (PFS)

The PFS is also known under the name INET, and this appears in UNIX System Services definitions when a FILESTYPE 1 and NETWORK TYPE 5 need to be defined in the BPXPRMxx member of SYS1.PARMLIB (Figure 2-7 on page 27).

Common INET Physical File System (CINET)

If you have two or more AF_INET transport providers on an MVS image (such as a production TCP/IP stack together with a test TCP/IP stack, or a TCP/IP stack with an AnyNet stack) you must use the Common INET Physical File System. Figure 2-6 shows a multiple stack environment with Common INET.

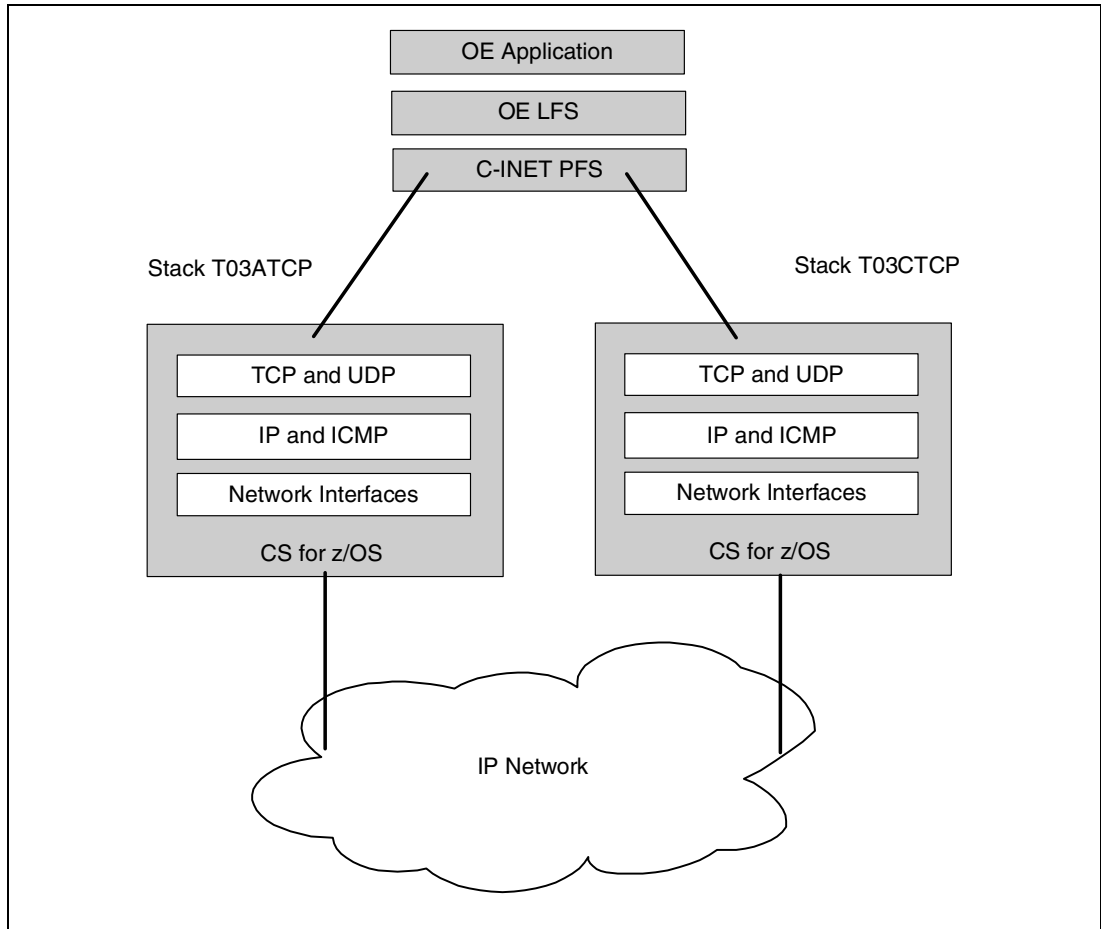


Figure 2-6 Multiple AF_INET transport providers: CINET PFS

SYS1.PARMLIB(BPXPRMxx) definitions for AF_INET

The following samples show BPXPARM definitions for single and multiple transport providers.

Integrated Sockets PFS definitions

Figure 2-7 shows the definitions for a single transport provider.

FILESYSTYPE	TYPE(INET) 1
	ENTRYPOINT(EZBPFINI) 2
NETWORK	DOMAINNAME(AF_INET) 3
	DOMAINNUMBER(2)
	MAXSOCKETS(10000) 4
	TYPE(INET) 5
	INADDRANYPORT(2000)
	INADDRANYCOUNT(2000)

Figure 2-7 BPXPRMxx for single stack using INET

2 EZBPFINI specifies INET with TCP/IP as transport provider.

3 AF_INET is the socket address type for this transport provider.

4 Parameter MAXSOCKETS is the maximum number of INET sockets that can be obtained. Ensure this number is large enough to accommodate all applications that may request a socket.

Common INET PFS definitions

Figure 2-8 on page 28 shows the BPXPRMxx definitions for the Common INET Physical File System.

```

FILESYSTYPE TYPE(CINET) 1
              ENTRYPOINT(BPXCINT) 2
NETWORK      DOMAINNAME(AF_INET)
              DOMAINNUMBER(2)
              MAXSOCKETS(5000)
              TYPE(CINET) 1
              INADDRANYPORT(2000)
              INADDRANYCOUNT(2000)

SUBFILESYSTYPE NAME(TCPIPA) 3
                TYPE(CINET) 1
                ENTRYPOINT(EZBPFINI) 4
                DEFAULT

SUBFILESYSTYPE NAME(TCPIPB) 3
                TYPE(CINET) 1
                ENTRYPOINT(EZBPFINI) 4

SUBFILESYSTYPE NAME(TCPIPC) 3
                TYPE(CINET) 1
                ENTRYPOINT(EZBPFINI) 4

SUBFILESYSTYPE NAME(ANYNET03) 3
                TYPE(CINET) 1
                ENTRYPOINT(ISTOEPIT) 4

```

Figure 2-8 Sample BPXPRM definition for Common INET (multiple stacks)

In this example, the TYPE value is now CINET 1 with its ENTRYPOINT(BPXCINT) 2

A transport provider stack for CINET is specified with a SUBFILESYSTYPE 3 statement with its ENTRYPOINT 4. The NAME keyword contains the started task name of this transport provider stack (TCPIPA, TCPIPB, TCPIPC, or the AnyNet stack ANYNET03).

Summary of BPXPRMxx definitions

In tabular form, we give you a summary of the BPXPRMxx definitions: first for a single-stack environment (Table 2-1) and then for a multi-stack system (Table 2-2).

Table 2-1 BPXPRMxx for a single OE transport provider

Transport Provider	FILESYSTYPE TYPE	FILESYSTYPE ENTRYPOINT	NETWORK TYPE	SUBFILESYSTYPE TYPE	SUBFILESYSTYPE ENTRYPOINT
TCP/IP	INET	EZBPFINI	INET	n/a	n/a
AnyNet	INET	ISTOEPIT	INET	n/a	n/a

Table 2-2 BPXPRMxx for multiple OE transport providers

Transport Provider	FILESYSTYPE TYPE	FILESYSTYPE ENTRYPOINT	NETWORK TYPE	SUBFILESYSTYPE TYPE	SUBFILESYSTYPE ENTRYPOINT
TCP/IP	CINET	BPXTCINT	CINET	CINET	EZBPFINI
AnyNet	CINET	BPXTCINT	CINET	CINET	ISTOEPIT

2.4 Customization of UNIX System Services

The customization of UNIX System Services is well explained in the *z/OS V1R2.0 UNIX System Services Planning*, SA22-7800.

TCP/IP requires UNIX System Services to be running in *full-function mode*. The following sections detail UNIX System Services customization that is relevant to the TCP/IP implementer.

2.4.1 Started task user IDs

The UNIX System Services tasks OMVS and BPXOINIT need the special user ID OMVSKERN assigned to them. OMVSKERN has to be defined as superuser with UID 0, program /bin/sh and home directory /.

TCP/IP tasks need RACF user IDs with the OMVS segment defined. The user ID associated with the main TCP/IP address space must be defined as a superuser; the requirements for the individual servers vary but most need to be a superuser as well.

2.4.2 Parmlib definitions

We show the IEASYSxx parmlib definitions that we used and explain the statements that are relevant to TCP/IP.

```

.....
OMVS=03, 1
SMS=02, 2
.....

```

Figure 2-9 IEASYSxx indicating BPXPRM03

In Figure 2-9, 1 specifies BPXPRM03 to be used, showing that someone customized OMVS.

2 specifies IGDSMS02 to be used, showing that someone customized SMS. We will not talk more about SMS.

The next two figures show our BPXPRM03 member of SYS1.PARMLIB.

```
MAXPROCSYS (300)
MAXPROCUSER (10125) 5
MAXUIDS (500)
MAXFILEPROC (65535)
MAXPTY (256)
MAXRTYS (256)
CTRACE (CTIBPX00)
FILESYSTYPE TYPE (HFS)
                ENTRYPOINT (GFUAINIT)
FILESYSTYPE TYPE (IBMUDES)
                ENTRYPOINT (BPXTUINIT)
NETWORK DOMAINNAME (AF_UNIX)
          DOMAINNUMBER (1)
          MAXSOCKETS (64)
          TYPE (IBMUDES)

FILESYSTYPE TYPE (CINET) 1
                ENTRYPOINT (BPXTCINT)
NETWORK DOMAINNAME (AF_INET)
          DOMAINNUMBER (2)
          MAXSOCKETS (10000)
          TYPE (CINET) 1
          INADDRANYPORT (4500)
          INADDRANYCOUNT (1500)

SUBFILESYSTYPE NAME (TCPIPA) 3
                TYPE (CINET) 1
                ENTRYPOINT (EZBPFINI) 4
                DEFAULT

SUBFILESYSTYPE NAME (TCPIPB) 3
                TYPE (CINET) 1
                ENTRYPOINT (EZBPFINI) 4

SUBFILESYSTYPE NAME (TCPIPC) 3
                TYPE (CINET) 1
                ENTRYPOINT (EZBPFINI) 4

SUBFILESYSTYPE NAME (ANYNET03) 3
                TYPE (CINET) 1
                ENTRYPOINT (ISTOEPIT) 4
```

Figure 2-10 BPXPRM03 part 1

1 in Figure 2-10 shows how a multi-stack environment has probably been planned for, since CINET is coded throughout.

3 are the names of the TCP/IP started tasks that are to run. With OS/390 V2R5 IP or later, it reflects the started task job name.

4 is the entry point that must be associated with TCP/IP or AnyNet.

If you are using the OMVS default user, the default value for MAXPROCUSER **5** will be too low. You have to estimate how many users will use the default user at the same time. The default user is typically used with FTP. In our case, we have changed the default value to 10125, clearly an overestimation, but also a guarantee that we will not reach the limit. Setting this value to something so high, however, may result in wasted storage, so one should try to not reach the limit, but not overestimate too much.

```

/***** /
  ROOT      FILESYSTEM('OMVS.SA03.ROOT') 1
            TYPE(HFS)
            MODE(RDWR)
/***** /
MOUNT FILESYSTEM('OMVS.SA03.ETC') 2
      MOUNTPOINT('/etc')
      TYPE(HFS)
      MODE(RDWR)

MOUNT FILESYSTEM('OMVS.SA03.TMP') 3
      MOUNTPOINT('/tmp')
      TYPE(HFS)
      MODE(RDWR)
MOUNT FILESYSTEM('OMVS.SA03.USER') 4
      MOUNTPOINT('/u')
      TYPE(HFS)
      MODE(RDWR)
MOUNT FILESYSTEM('SMS.OMVS.KARL') 5
      TYPE(HFS)
      MODE(RDWR)
      MOUNTPOINT('/u/karl')

      .....

MOUNT FILESYSTEM('SMS.OMVS.KAKKY') 5
      TYPE(HFS)
      MODE(RDWR)
      MOUNTPOINT('/u/kakky')

```

Figure 2-11 BPXPRM03 part 2

1 in Figure 2-11 shows the name of the root file system that has been created for the UNIX System Services installation. The procedure for creating a UNIX System Services system is detailed in the *z/OS V1R2.0 UNIX System Services Planning, SA22-7800* and in the *z/OS V1R2.0 Program Directory, Program Number 5694-A01, G110-0670*.

2 indicates the HFS directory and file system that will be used for configuration files.

3 shows a separate file system for the /tmp directory. Files in /tmp can potentially become very large.

4 shows the parent directory for the user files. This file system could be combined with the user file systems **5** depending on individual installation preferences.

5 shows a series of user directories and file systems that are to be mounted for individual OMVS users.

Another way to mount the user file systems from BPXPRMxx is to use the AUTOMOUNT facility. This facility mounts the file systems only when needed and unmounts them if they are not used for a specific amount of time, allowing management of the file systems. The AUTOMOUNT facility is described in the *z/OS V1R2.0 UNIX System Services Planning*, SA22-7800.

Note: For the initial startup of OMVS only the root file system is required.

On our system all program products have been installed into the root file system. This might be different on your system. You could have all products installed in separate file systems. If that is the case, they have to be included in BPXPRMxx as shown in the following example:

```
MOUNT FILESYSTEM('OMVS.&SYSNAME..TCP/IP')
      MOUNTPOINT('/usr/lpp/tcpip')
      TYPE(HFS)  MODE(RDWR)
```

For the /tmp directory there is also the option to put it into main storage for better performance and space management. To do this, you have to use the following statements in BPXPRMxx:

```
FILESYSTYPE TYPE(TFS) ENRYPPOINT(BPXTFS)
MOUNT FILESYSTEM('/TMP')
      TYPE(TFS)  MODE(RDWR)
      MOUNTPOINT('/tmp')
      PARM('-s 500')
```

2.4.3 OMVS start-up at IPL time

If you IPL your z/OS system with PARMLIB definitions similar to ours you should get the messages shown in Figure 2-12. Note how messages issued by z/OS UNIX begin with the prefix *BPX*.


```

IEE252I MEMBER BPXPRM03 FOUND IN SYS1.PARMLIB      1
IEF196I      1 //OMVS      JOB MSGLEVEL=1
IEF196I      2 //STARTING EXEC OMVS
IEF196I      2 IEFC001I PROCEDURE OMVS WAS EXPANDED USING SYSTEM
IEF196I LIBRARY SYS1.PROCLIB
IEF196I      3 XXOMVS PROC
IEF196I      4 XXOMVS EXEC      PGM=BPXINIT,REGION=OK
IEE252I MEMBER CTIBPX00 FOUND IN SYS1.PARMLIB

.....

IEE252I MEMBER IGDSMS02 FOUND IN SYS1.PARMLIB      2
IEE536I SMS      VALUE 02 NOW IN EFFECT
IGD020I SMS IS NOW ACTIVE                          3
IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS00001
BPXF013I FILE SYSTEM OMVS.SA03.ROOT                4
WAS SUCCESSFULLY MOUNTED.

.....

IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS00008
BPXF013I FILE SYSTEM SMS.OMVS.GDENTE
WAS SUCCESSFULLY MOUNTED.
BPXF203I DOMAIN AF_UNIX WAS SUCCESSFULLY ACTIVATED. 5
BPXF203I DOMAIN AF_INET WAS SUCCESSFULLY ACTIVATED. 5
IEF196I      1 //BPX0INIT JOB MSGLEVEL=1           6
IEF196I      2 //STARTING EXEC BPX0INIT
IEF196I      2 IEFC001I PROCEDURE BPX0INIT WAS EXPANDED USING SYSTEM
IEF196I LIBRARY SYS1.PROCLIB
IEF196I      3 XXBPX0INIT PROC
IEF196I      4 XXBPX0INIT EXEC PGM=BPXPINPR,REGION=OK,TIME=NOLIMIT
IEF403I BPXAS - STARTED 7

.....

BPXI004I OMVS INITIALIZATION COMPLETE              8

```

Figure 2-12 Initialization of OMVS (UNIX System Services)

At 1 in Figure 2-12 you see that MVS found the defined OMVS PARMLIB member and starts OMVS. 2 shows the start of SMS initialization. After SMS has initialized at 3, OMVS continues with the mount of the file systems 4. At 5 OMVS has successfully activated the network domains and starts the initialization process 6. When the initialization process forks child processes, WLM starts BPXAS address spaces 7 as needed. Finally OMVS has initialized 8.

2.4.4 OMVS displays

You can issue the command **D SMS** to verify that the system is running a functional SMS environment.

```

D SMS
IGD002I 18:18:42 DISPLAY SMS 687
SCDS = IP01.DFSMS.SCDS
ACDS = IP01.DFSMS.ACDS
COMMDS = SYSPLEX.COMMDS
DINTERVAL = 150
REVERIFY = NO
ACSDEFAULTS = NO
  SYSTEM      CONFIGURATION LEVEL      INTERVAL SECONDS
  RA03        2000/09/06 18:18:28        15
  RA28        2000/09/06 18:18:28        15
  RA39        2000/09/06 18:18:20        15

```

Figure 2-13 SMS display

You see in the SMS display that we have executed the command from a sysplex environment and have captured information pertaining to both SA03 and SA28. In Figure 2-14, you can see that the OMVS member that is running is related to BPXPRM03 **1** and that the initialization process **2** is running as superuser OMVSKERN **3**.

```

D OMVS,ASID=ALL
BPX0040I 11.54.06 DISPLAY OMVS 825
OMVS      000E ACTIVE      OMVS=(03) 1
USER      JOBNAME ASID      PID      PPID STATE  START  CT_SECS
OMVSKERN  BPXOINIT 0022      1        0 MFI   13.33.56  40.066
3      2

```

Figure 2-14 Running the OMVS system

What is significant here is that OMVS=DEFAULT is not displayed in the output. Recall from 2.1, “Customization levels of UNIX System Services” on page 17 that UNIX System Services must be customized in *full-function mode*. The display tells you that, at the very least, your system is not running in default mode (*minimal mode*).

Figure 2-15 shows a display of the available file systems. This display should list all mount statements in your BPXPARMxx member.

```

D OMVS,F
BPX0045I 11.59.06 DISPLAY OMVS 883
OMVS      000E ACTIVE          OMVS=(03)
TYPENAME  DEVICE -----STATUS-----  MODE
.....
HFS              6 ACTIVE                      RDWR
  NAME=OMVS.SA03.ETC
  PATH=/etc
HFS              5 ACTIVE                      RDWR
  NAME=OMVS.SA03.USER
  PATH=/u
HFS              4 ACTIVE                      RDWR
  NAME=OMVS.SA03.TMP
  PATH=/SYSTEM/tmp
HFS              3 ACTIVE                      RDWR
  NAME=HFS.RA03.010RB1.ROOT
  PATH=/

```

Figure 2-15 Display of mounted file systems

2.5 Working with UNIX System Services: interactive interfaces for the end user

As you have read in the overview of UNIX System Services, there are a couple of ways to work interactively with UNIX System Services:

1. You can display information about OMVS from the MVS console.
2. After you have configured and started TCP/IP you can use telnet and rlogin to access UNIX System Services.
3. You may enter commands from the z/OS UNIX shell. You enter the shell by specifying **OMVS** from TSO, and you are presented with a screen similar to the one in Figure 2-16.
4. You may enter commands from the ISHELL environment, depicted in Figure 2-17.

```

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.

- - - - -
- Improve performance by preventing the propagation -
- of TSO/E or ISPF STEPLIBs -
- - - - -
ADOLFO @ RA03:/u/adolfo>

====>

                                INPUT
ESC=¢  1=Help      2=SubCmd   3=HlpRetrn  4=Top      5=Bottom   6=TSO
        7=BackScr  8=Scroll  9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve

```

Figure 2-16 OMVS shell

This shell provides a command interface to the UNIX System Services environment. The OMVS shell, licensed from the Mortice Kern company, is XPG4- and POSIX 1003.2-compliant.

Another way to communicate with TCP/IP is to interface with the UNIX environment by means of an ISPF interface called the ISHELL with the TSO commands **ISHELL** or **ISH**. Figure 2-17 displays the ISHELL environment.

```

File Directory Special_file Tools File_systems Options Setup Help
-----
OpenMVS ISPF Shell

Enter a pathname and do one of these:

- Press Enter.
- Select an action bar choice.
- Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
More:      +

/u/adolfo
_____
_____
_____

(C) Copyright IBM Corp., 1993,2000. All rights reserved.

Command ==> _____
F1=Help    F3=Exit    F5=Retrieve F6=Keyshelp F7=Backward F8=Forward
F10=Actions F11=Command F12=Cancel

```

Figure 2-17 ISHELL interface

The ISHELL can be used conveniently by users and system administrators to enter commands for the TCP/IP stack or to perform activities such as creating files, assigning UIDs to users, and so on.

2.5.1 Displaying OMVS processes

You can display the processes running in OMVS from the MVS console or from any of the shell interfaces.

Figure 2-18 shows the list of processes with their associated user IDs.

```

D OMVS,A=ALL
BPX004OI 12.14.02 DISPLAY OMVS 981
OMVS 000E ACTIVE OMVS=(03)
USER JOBNAME ASID PID PPID STATE START CT_SECS
OMVSKERN BPXOINIT 0022 1 0 MF 13.33.56 40.344
 LATCHWAITPID= 0 CMD=BPXPINPR
 SERVER=Init Process AF= 0 MF=00000 TYPE=FILE
TCPIPA TCPIPA 0040 16777222 1 MR 13.34.36 701.078
 LATCHWAITPID= 0 CMD=EZBTCPIP
OMVSKERN SYSLOGD1 003B 16777223 1 1FI 13.34.12 10.877
 LATCHWAITPID= 0 CMD=/usr/sbin/syslogd -f /etc/syslog.conf
TCPIPA TCPIPA 0040 8 1 1R 13.34.41 701.078
 LATCHWAITPID= 0 CMD=EZBTSSL
TCPIPA TCPIPA 0040 9 1 1R 13.34.42 701.078
 LATCHWAITPID= 0 CMD=EZBTMCTL
TCPIPA TCPIPA 0040 10 1 1F 13.34.42 701.078
 LATCHWAITPID= 0 CMD=EZACFALG
TCPIPA TCPIPA 0040 11 1 1F 13.34.44 701.078
 LATCHWAITPID= 0 CMD=EZASASUB
...
KAKKY KAKKY 0052 33554541 1 MRI 09.30.31 79.618
 LATCHWAITPID= 0 CMD=EXEC
PABST PABST 003C 33554552 1 MRI 11.11.51 10.459
 LATCHWAITPID= 0 CMD=EXEC
ADOLFO ADOLFO 0028 83886238 15 1FI 18.53.45 .510
 LATCHWAITPID= 0 CMD=otelnetsd -Y 9.24.106.51 -p adolfo -a dum

```

Figure 2-18 UNIX System Services processes display from the MVS console

In Figure 2-18, multiple tasks are associated with the same RACF user ID, TCPIPA. This has the advantage of easier maintenance but the disadvantage of no distinguishing features among messages for individual tasks. Many users of TCP/IP and UNIX System Services would lean towards ease of problem determination and would assign individual RACF user IDs to each OMVS user.

You will find a thorough discussion of RACF authority and superuser in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)).

Another way to look at UNIX processes is to use the shell environment and to execute the UNIX command `ps -ef`. This displays all processes and their environments in forest or family tree format:

```

ADOLFO @ RA03:/u/adolfo>ps -ef
# ps -ef
OMVSKERN      1          0 - Sep 05 ?      1:01 BPXPINPR
OMVSKERN 16777218      1 - Sep 05 ?      0:00 BPXVCLNY
OMVSKERN 83886083      1 - Sep 05 ?      0:00 BPXVCMT
OMVSKERN 50331653      1 - Sep 05 ?      0:00 GFSCMAIN
OMVSKERN 16777222      1 - Sep 05 ?     17:22 EZBTCPIP
OMVSKERN 16777223      1 - Sep 05 ?      0:16 /usr/sbin/syslogd -f /
etc/syslog.conf
OMVSKERN      8          1 - Sep 05 ?     17:22 EZBTTSSL
OMVSKERN      9          1 - Sep 05 ?     17:22 EZBTMCTL
OMVSKERN     10          1 - Sep 05 ?     17:22 EZACFALG
OMVSKERN     11          1 - Sep 05 ?     17:22 EZASASUB
OMVSKERN 16777228      1 - Sep 05 ?     15:59 /usr/lpp/tcpip/sbin/om
proute
....
KAKKY 33554541          1 - Sep 06 ?      1:20 EXEC
4011 83886193 16777414 - Sep 07 ttyp0002 0:01 sh -L
PABST 33554552          1 - Sep 06 ?      0:10 EXEC
...
ADOLFO @ RA03:/u/adolfo>

```

Figure 2-19 UNIX System Services processes display from the shell

Notice that in Figure 2-19, the processes running with a superuser are shown as running with user ID OMVSKERN. The reason for this is that RACF cannot map a UNIX System Services UID to an MVS user ID correctly if there are multiple MVS user IDs defined with the same UID. So RACF uses the last referenced MVS user ID.

2.5.2 Working with file systems

You can display mounted file systems from ISHELL or the MVS console. Figure 2-20 shows the console display and Figure 2-21 the display from ISHELL.

```

D OMVS,F
BPX0045I 15.22.32 DISPLAY OMVS 474
OMVS 000E ACTIVE OMVS=(03)
TYPENAME DEVICE -----STATUS----- MODE
...
HFS 6 ACTIVE RDWR
NAME=OMVS.SA03.ETC
PATH=/etc
HFS 5 ACTIVE RDWR
NAME=OMVS.SA03.USER
PATH=/u
HFS 4 ACTIVE RDWR
NAME=OMVS.SA03.TMP
PATH=/SYSTEM/tmp
HFS 3 ACTIVE RDWR
NAME=HFS.RA03.010RB1.ROOT
PATH=/

```

Figure 2-20 Displaying mounted filesystems from MVS console

From ISHELL you can get more information and can manipulate the file systems.

```
Work with Mounted File Systems

Select one or more file systems with / or action codes.
  U=Unmount  A=Attributes  C=Change mode  R=Reset unmount or quiesce
  File system name          Status          Row 1 of 10
_ OMVS.SA03.ROOT           Available
_ OMVS.SA03.ETC            Available
_ OMVS.SA03.TMP            Available
_ OMVS.SA03.USER           Available
A SMS.OMVS.GIANCA          Available
_ SMS.OMVS.GDENTE          Available
_ SMS.OMVS.KAKKY           Available
_ SMS.OMVS.KARL            Available
```

Figure 2-21 Viewing the mount table

From the mount table you can display the file system attributes.

```
File System Attributes

File system name:
OMVS.RA03.U.GIANCA
Mount point:
/u/gianca

More:      +

Status . . . . . : Available
File system type . . . : HFS
Mount mode . . . . . : R/W
Device number . . . . . : 11
Type number . . . . . : 1
DD name . . . . . : SYS00011
Block size . . . . . : 4096
Total blocks . . . . . : 6840
Available blocks . . . : 1492
Blocks in use . . . . . : 5319

F1=Help      F3=Exit      F4=Name      F6=Keyshelp
F12=Cancel
```

Figure 2-22 File system attributes

The next screens show you how to mount a file system. To do this, you have to be a superuser with RACF OPERATIONS authority.


```

File Directory Special_file Tools File_systems Options Setup Help
- +-----+
- |                                     |
- |                               Mount a File System                       |
- |                                     |
E | Mount point:                               More:                          |
- |                                     |
- |   /u/gdente                               |
- | _____|
- | _____|
R |                                     |
- | File system name . . sms.omvs.gdente_____|
- | File system type . . hfs_____|
- | New owner . . . . . _____|
- |                                     |
- | Select additional mount options:    |
- | _ Read-only file system              |
- | _ Ignore SETUID and SETGID           |
- | Mount parameter:                    |
- | _____|
- | F1=Help    F3=Exit    F4=Name    F6=Keyshelp  F12=Cancel |
- | _____|
+ +-----+

```

Figure 2-23 Mount file systems

2.5.3 Manipulating files and directories

Figure 2-24 and Figure 2-25 show the steps taken in the ISHELL to define the HFS home directory `/u/gdente` for the user GDENTE.

```

File Directory Special_file Tools File_systems Options Setup Help
-----+-----+-----+
Enter | 2 1. List directory(L)...      | Shell
      | 2. New(N)...                  |
      | 3. Attributes(A)...          |
-   | 4. Delete(D)...              |
-   | 5. Rename(R)...              |
-   | 6. Copy to PDS(C)...         |
      | 7. Copy from PDS(I)...       | n the command line.
Retur | 8. Print(P)                   |
      | 9. Compare(M)...             | rent pathname.
/u    | 10. Find strings(F)...        | More:
     _ | 11. Set working directory(W) |
     _ | 12. File system(U)...        |
-----+-----+-----+

```

Figure 2-24 Superuser in ISHELL creating `/u/gdente/` user directory #1

Be sure that the UNIX permission bits have been set for Read, Write, and Executable modes.

```

File Directory Special_file Tools File_systems Options Setup Help
- + -----+
|                                     |
|           Enter File Permissions   |
|                                     |
| E | Permissions . . 755 (3 digits, each 0-7) | |
|   |                                     |
|   | F1=Help           F3=Exit           F6=Keyshelp |
|   | F12=Cancel       |                                     |
|   |                                     |
| - + -----+
|
| Return to this panel to work with a different pathname.
|
|                                     More:
|
| /u/gdente/
| _____
| _____
| _____

```

Figure 2-25 Creating user directory #2

From the OMVS shell you would just enter `mkdir /u/gdente`. In this case the permission bits would be set as specified in `/etc/profile` or `$home/.profile`. You want to verify them with the `ls -a11` command and if you do not like them you have to change them with the `chmod` command.

If you have HFS files, you can browse them using the ISHELL tools or you can execute the `obrowse` command as follows from the OMVS shell environment:

```

# obrowse /tmp/ddns.dat

***** Top of Data *****
*.small.isp.com mvs03.itso.ral.ibm.com Hd1CgHs5eMgGtXu050Rn98n0i2dn0z8FEEd
*.100.168.192.in-addr.arpa mvs03.itso.ral.ibm.com 4jDkIDnrU4Ii9WyqvGcx9y/
100.168.192.in-addr.arpa mvs03.itso.ral.ibm.com Y3BU88qXZQOE0ryPqC1n9+vxN
small.isp.com mvs03.itso.ral.ibm.com mJw7G8UfJAuUaKgk3A1D97VM5757d6c5sY62
***** Bottom of Data *****

```

Figure 2-26 obrowse example

To edit files, you can use the ISHELL tools or you can use the `oedit` command from the OMVS shell.

Note: Both `obrowse` and `oedit` are TSO commands. If you used telnet or rlogin to get to the UNIX System Services shell, you have to use the `cat` command and the vi editor.

All methods have advantages and it is a matter of personal preference which one you use. The ISHELL provides an ISPF look and feel, the OMVS shell a more UNIX or DOS look and feel, and of course for real UNIX users there is the vi editor.

2.5.4 Superuser mode

Certain commands and operations from OMVS or from the ISHELL are authorized only for superusers. There are two alternatives for running as a superuser:

1. The user ID may have permanent superuser status. This means that the ID has been created with a UID value of 0.
2. The user ID may have temporary authority for the superuser tasks. The defined UID will have been set up as a non-zero value in RACF, but the user will have been granted READ access to the RACF facility class of BPX.SUPERUSER. Also, RACF provides superuser granularity enhancements to assign functions to users that need them.

If you need only temporary authority to enter superuser mode, then the granting of simple READ permission to the BPX.SUPERUSER facility class will allow the user to switch back and forth between superuser mode and standard mode. You enter `su` from the OMVS shell, as in Figure 2-28, or you may select SETUP OPTIONS from the ISHELL and specify Option #7 to obtain superuser mode.

```

File Directory Special_file Tools File_systems Options Setup Help
-----+-----+
                                OpenMVS ISPF S _7_ 1. *User...
Enter a pathname and do one of these:                2. *User list...
- Press Enter.                                       3. *All users...
- Select an action bar choice.                       4. *All groups...
- Specify an action code or command on              5. *Permit field access...
                                                    6. *Character Special...
                                                    7. Enable superuser mode(SU)
Return to this panel to work with a diffe +-----+
/u/gdente                                           Some choices (*) require
                                                    superuser or the "special"
                                                    attribute for full function, or
                                                    both

```

Figure 2-27 Obtaining superuser mode

At this point the user can enter commands authorized for the superuser function from the ISHELL, or he can switch to an OMVS shell he has already signed onto. The original OMVS shell is displayed in Figure 2-28. Notice the basic prompt level, indicated by the `$` prompt (1).

The user is still in basic user mode when he queries his identity (2). Next the user executes a command from the OMVS shell to put himself in superuser mode (3). The prompt sign changes to `#` (4). The user queries his identity once more and discovers his identity has changed to the OMVSKERN (5). The `exit` command (6) takes him out of superuser mode and back into the basic user mode. A final exit (7) will return the user to TSO once he presses Enter.

```

$                               1
$ whoami                         2
GDENTE
$ su                              3
# whoami                         4
OMVSKERN                         5
# exit                           6
$ exit                           7

>>>> FSUM2331 The session has ended. Press <Enter> to end OMVS. OS/390 Firewall Technolo
Guide and Reference

```

Figure 2-28 In the OMVS shell: alternating superuser and non-superuser modes

2.6 Common user errors with UNIX System Services

In this section, we show some simple problems that we encountered.

2.6.1 Problems with the home directory

In the next example, the TSO user attempted unsuccessfully to enter the OMVS shell interface from ISPF; the user has an OMVS segment defined but another problem occurs.

```

Menu List Mode Functions Utilities Help
-----
                                ISPF Command Shell
Enter TSO or Workstation commands below:

===> omvs

Place cursor on choice and press enter to Retrieve command

=>
FSUM2078I No session was started. The home directory for this TSO/E
user does not exist or cannot be accessed.+
FSUM2079I Function = sigprocmask, return value = FFFFFFFF, return code
code = 9C reason code = 0507014D
***

```

Figure 2-29 Executing OMVS command

A similar problem occurs when trying to access the ISHELL environment. See Figure 2-30.

```

File Directory Special_file Tools File_systems Options Setup Help
+-----+
|                                     |
|                               Make a File System                               |
|-----|
| File system name . . . . . _____|
| Primary cylinders . . . . . _____|
| Secondary cylinders . . . . . _____|
| Storage class . . . . . _____|
| Management class . . . . . _____|
| Data class . . . . . _____|
|-----|
| F1=Help      F3=Exit      F6=Keyshelp  F12=Cancel|
|-----|
|-----|
| Errno=9Cx Process Initialization error; Reason=0507014D The dub fail|
| due to an error with the initial home directory. Press Enter to|
| continue.                                         |
|-----|
+-----+

```

Figure 2-30 Executing ISHELL command

In both cases the user had an OMVS segment defined for him in RACF. However, the UNIX System Services implementer had neglected to define or authorize the home directory that had been associated with the user in his OMVS segment. (You can see what this home directory is supposed to be with the RACF command `listuser`.) Authorization is provided with the permission bits.

Note: If your system is OS/390 V2R7 or later, you will get access to the HFS even if you cannot access your home directory and get the above message.

The same symptom shows up for users without an OMVS segment defined if the BPX.DEFAULTUSER facility has been activated with an inaccessible home directory.

2.6.2 UNIX permission bits

You have already read something about setting up appropriate UNIX permission bits. Figure 2-31 shows an example of incorrect permission bits set for a user.

```

ICH408I USER(GDENTE ) GROUP(WTCRES ) NAME(GWEN DENTE ) 703
/u/gdente CL(DIRSRCH ) FID(01E2D7D3C5E7F34E2B0F000000000003)
INSUFFICIENT AUTHORITY TO LOOKUP
ACCESS INTENT(--X) ACCESS ALLOWED(OTHER ---)
ICH408I USER(GDENTE ) GROUP(WTCRES ) NAME(GWEN DENTE ) 704

```

Figure 2-31 Errors with UNIX permission bit settings

Although the user had the UNIX permission bit settings of 755 on the `/u/gdente/` directory, the permission bits were set at 600 for the `/u/` directory as shown in Figure 2-32. The moral of this story is to ensure that all directories in the entire path are authorized with suitable permission bits. When the settings were changed to 755 for the `/u/` directory, access to the subdirectory was allowed.

You may display UNIX permission bits from the ISHELL environment or by issuing the command `ls -a1F` from the shell:

```
# cd /u/gdente
# ls -a1F
total 32
drwxr-xr-x  2 GDENTE  OMVSGRP      0 Jan 28 17:56 ./
drw-----  7 WOZABAL  SYSPROG      0 Jan 28 18:03 ../
```

Figure 2-32 Permission bit display

The `-a1F` options indicate that all files should be listed, including hidden files, that the long format should be displayed, and that the flags about the type of file (link, directory, etc.) should be given.

2.6.3 Default search path and symbolic links

The directory search path is specified in the environment variable `$PATH`. Normally this environment variable is set system-wide in `/etc/profile` and can be further customized for individual users in `$home/.profile`. The sample for `/etc/profile` sets `$PATH` to:

```
/bin:.
```

and should be expanded to:

```
/bin:/usr/sbin:.
```

or:

```
./bin:/usr/sbin
```

depending on whether you want the current directory searched first or last. The instructions for setting up this user profile are contained in *z/OS V1R2.0 UNIX System Services User's Guide, SA22-7801* and *z/OS V1R2.0 UNIX System Services Planning, SA22-7800*.

Note: To view the search path that has been established for you, issue `echo $PATH` from the shell environment.

A user may attempt to run a simple TCP/IP command such as `oping` and receive an error that the command is not found:

```
BROWSE -- /tmp/GDENTE.16:13:33.712714.ishell ---
Command ==>
***** Top of Data ***
oping: FSUM7351 not found
***** Bottom of Data *
```

The user then finds he must preface the command with the directory path necessary to locate it:

```
/usr/lpp/tcpip/bin/oping
```

If you experience such a problem, check that the symbolic links are correct. Part of the installation is to run the OE MKDIR program to set up the symbolic links for the various commands and programs from their real path to /bin or /usr/sbin, where they can be found using the default search path.

2.6.4 Incorrect RESOLVER_CONFIG in use

Every client process requires an affinity to one or more TCP/IP stacks. Traditionally the affinity is established through pointers to the TCPIP.DATA data set. With UNIX System Services integrated into Communications Server for z/OS IP, a TSO client looking for TCPIP.DATA information may use one search path and a UNIX shell client may use a different one. You may find that a process works one way when executed from TSO and another way when executed from the shell; this may occur if a different TCPIP.DATA file is found for each client type.



Installation

Installation and customization of CS for z/OS IP consists of the following four major steps:

- ▶ Planning
- ▶ Preinstallation
- ▶ Installation
- ▶ Product customization

This chapter provides examples and usage guidelines on how the product can be installed. We also give you an introduction to the basic configuration tasks for CS for z/OS IP; more customization tasks are explained in subsequent chapters.

This chapter contains the following sections:

- ▶ 3.1, “First things first” on page 49
- ▶ 3.2, “Planning your installation and migration” on page 50
- ▶ 3.3, “Preinstallation” on page 51
- ▶ 3.4, “Security considerations” on page 53
- ▶ 3.5, “Installation” on page 59
- ▶ 3.6, “Message types: Where to find them” on page 66
- ▶ 3.7, “Checklist for installation and customization” on page 67

3.1 First things first

Before installing the Communications Server for z/OS, you should first review the *z/OS V1R2.0 Program Directory, Program Number 5694-A01, GI10-0670*. Review all the installation requirements and considerations before starting with the actual installation. Finally, we need to include a summary of important installation and customization points in the form of a checklist.

3.2 Planning your installation and migration

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

1. *z/OS V1R2.0 Program Directory, Program Number 5694-A01, GI10-0670*
2. Preventive Service Planning (PSP) bucket
3. The z/OS Web pages:
<http://www-1.ibm.com/servers/eserver/zseries/zos/installation/installz12.html>
4. *z/OS V1R2.0 CS: IP Migration, GC31-8773*
5. *z/OS V1R2.0 UNIX System Services Planning, SA22-7800*
6. z/OS V1R2 Installation Planning Wizard located at:
<http://www-1.ibm.com/servers/eserver/zseries/zos/wizards/ipw/ipwv1r2/>

Planning for and installing Communications Server for z/OS IP requires MVS, UNIX, and networking skills. If your background is in traditional MVS programming or systems programming, the UNIX System Services terminology may at first seem to be somewhat confusing. If your background is in the UNIX environment, the same could be equally true for MVS terminology.

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

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

We discovered with the first version of a full TCP/IP stack based on native MVS and on UNIX System Services that very few people have all the requisite skills to implement CS for z/OS IP on their own successfully. As more and more systems programmers acquire skills in UNIX System Services and in TCP/IP, this will become less and less the case. Teaming with the UNIX System Services implementer when implementing CS for z/OS IP provides the best guarantee of overcoming the initial inhibitors to establishing a working CS for z/OS IP environment.

If you are migrating to Communications Server for z/OS, establish a migration process to move all your existing applications, and after this, consider the use of new and enhanced functions outlined in *z/OS V1R2.0 CS: IP Migration, GC31-8773*.

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

3.3 Preinstallation

When you install and customize Communications Server for z/OS IP, it will be very helpful to have the following at your immediate disposal:

- ▶ The migration plan, fallback plans, and test plans that you have customized for your company's implementation
- ▶ Printouts of procedures and data sets that you will be using for the implementation
- ▶ *z/OS V1R2.0 Program Directory, Program Number 5694-A01, GI10-0670*
- ▶ *z/OS V1R2.0 CS: IP Migration, GC31-8773*
- ▶ *z/OS V1R2.0 CS: IP Configuration Guide, SC31-8775*
- ▶ *z/OS V1R2.0 CS: IP Configuration Reference, SC31-8776*
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 1 (EZA), SC31-8783*
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 2 (EZB), SC31-8784*
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 3 (EZY), SC31-8785*
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 4 (EZZ-SNM), SC31-8786*
- ▶ *z/OS V1R2.0 CS: IP and SNA Codes, SC31-8791*
- ▶ *z/OS V1R2.0 UNIX System Services Planning, SA22-7800*
- ▶ *z/OS V1R2.0 UNIX System Services User's Guide, SA22-7801*
- ▶ *z/OS V1R2.0 UNIX System Services Messages and Codes, SA22-7807*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 1 (ABA-AOM), SA22-7631*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 2 (ARC-ASA), SA22-7632*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 3 (ASB-BPX), SA22-7633*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 4 (CBD-DMO), SA22-7634*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 5 (EDG-GFS), SA22-7635*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 6 (GOS-IEA), SA22-7636*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 7 (IEB-IEE), SA22-7637*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 8 (IEF-IGD), SA22-7638*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 9 (IGF-IWM), SA22-7639*
- ▶ *z/OS V1R2.0 MVS System Messages, Vol 10 (IXC-IZP), SA22-7640*

You should base your installation activities on the above-mentioned sources. You may use this redbook as a source of additional information, but do not use it as your only source during installation and customization of Communications Server for z/OS IP.

3.3.1 IP Migration Guide

CS for z/OS IP is a complex product that offers many features. Review the *z/OS V1R2.0 CS: IP Migration*, GC31-8773 to understand the new functions, enhancements, changes, software and hardware requirements, and prerequisites for Communications Server for z/OS IP. Then select the TCP/IP services to be offered to the users and identify all prerequisites and incorporate them into your implementation plan. *z/OS V1R2.0 Program Directory, Program Number 5694-A01*, G110-0670 and the PSP bucket may contain different information that supersedes the information in the *z/OS V1R2.0 CS: IP Migration*, GC31-8773, so take the information from all the sources into account.

3.3.2 z/OS Program Directory

The z/OS Program Directory contains not only the installation instructions for Communications Server for z/OS IP, but it also supersedes the information in the *z/OS V1R2.0 CS: IP Migration*, GC31-8773.

Figure 3-1 shows the cover of the Program Directory included with z/OS V1R2.



Figure 3-1 Cover of the z/OS V1R2 Program Directory

3.3.3 Program support

Contact the IBM Support Center or use IBMLINK to get the Preventive Service Planning (PSP) upgrade information. This document contains the latest information on new functions and maintenance that you may wish to take advantage of in your installation of Communications Server for z/OS IP. Its information supersedes that in the *z/OS V1R2.0 CS: IP Migration*, GC31-8773 and the *z/OS V1R2.0 Program Directory, Program Number 5694-A01*, GI10-0670.

If you are asked to provide more information about the FMIDs, Component IDs (COMP IDs), or Component Names for which you require service, you will find this information in one of the appendixes of the *z/OS V1R2.0 Program Directory, Program Number 5694-A01*, GI10-0670.

3.4 Security considerations

Security is an important consideration for most MVS installations. TCP/IP has some built-in internal security mechanisms and relies on the services of an external security manager such as the IBM Resource Access Control Facility (RACF). The external security manager is called via the MVS System Authorization Facility (SAF) interface.

An external security manager is a requirement in the Communications Server for z/OS IP environment. As an online application, it is important that TCP/IP undergo security checks to eliminate possible security exposures. Some basic security concepts are included in the following sections, but for a more detailed explanation refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)).

3.4.1 APF authorization

The TCP/IP system program libraries must be APF authorized. Authorized Program Facility (APF) means that MVS built-in security may be bypassed by programs that are executed from such libraries. CS for z/OS IP data sets have to be protected with RACF. Special attention has to be given to the APF authorized libraries defined in PROGxx.

We used the LNKAUTH=LNKLST specification in SYSx.PARMLIB member IEASYSxx, which means that all libraries in the LNKLST concatenation will be APF authorized. If these libraries are accessed through STEPLIB or JOBLIB, they will not be APF authorized unless they have been specifically defined in the IEAAPFxx or PROGxx member.

SEZALINK is one library that *must* be made part of your LNKLST concatenation. Because of the LNKAUTH=LNKLST specification, it will be APF authorized when it is accessed through the LNKLST concatenation. The SEZALINK library holds the TCP/IP system code, used by both servers and clients.

In addition to the LNKLST libraries, there are some libraries that are not accessed through the LNKLST concatenation, but have to be APF authorized. The SEZATCP library holds the TCP/IP system code used by servers. The library is normally placed in the STEPLIB or JOBLIB concatenation, which is part of the server JCL.

The following libraries may have to be APF authorized, depending on the choices you make during the installation of z/OS:

SEZALPA This library holds the TCP/IP modules that must be made part of your system's LPA. If you choose to add the library name to your LPALSTxx member in SYSx.PARMLIB, you also have to make sure the library is APF authorized. If

you copy the load modules in the library to an existing LPALSTxx data set, you do not need to authorize the SEZALPA data set.

SEZADSIL This library holds the load modules used by the SNMP command processor running in the NetView address space. If you choose to concatenate this library to STEPLIB in the NetView address space, you may have to APF authorize it, if other libraries in the concatenation are already APF authorized.

Every APF-authorized online application may have to be reviewed to ensure that it matches the security standards of the installation. A program is "a well-behaved program" if:

- ▶ Logged-on users cannot access or modify system resources for which they are not authorized and never would be by the security rules of the installation. Or, in other words, the program has no *security hole* that allows unauthorized penetration and data modification.
- ▶ The program does not require any special credentials to be able to execute.

Or, in the case of RACF, the program does not need the RACF authorization attribute OPERATIONS for execution.

Note: User IDs with the RACF attribute OPERATIONS have ALTER access to all data sets in the system. The access authority to single data sets may be specifically lowered or excluded.

3.4.2 RACF environment

RACF is very flexible and can be set up and tailored to meet almost all security requirements of large enterprises. RACF can be set up to match the scope of:

- ▶ A totally *centralized* enterprise structure, which includes centralized RACF administration and security audit functions.
- ▶ A totally *decentralized* structure, with either centralized or decentralized RACF administration and security audit functions.

The scope of decentralization may be either:

- An enterprise online application
- Several enterprise online applications
- Parts of an enterprise

All RACF implementations are based on the following key elements:

- ▶ User IDs
- ▶ Groups
- ▶ RACF resources
- ▶ RACF profiles
- ▶ RACF facility classes
- ▶ The hierarchical owner principle, which is applicable for all RACF definitions of user IDs, groups and RACF resources

RACF implementation

The TCP/IP RACF implementation is done using the following steps:

- ▶ The centralized RACF administration with system-wide RACF authorization sets up the decentralized RACF environment for TCP/IP and defines a RACF group for use as the RACF resource owner group for the TCP/IP application. The owner of this group may be any other group or user ID in the RACF cloud. We chose the RACF default group SYS1, which always exists in all RACF installations.

- ▶ Delegate the RACF authority to administer the decentralized TCP/IP RACF environment to the TCP/IP administration user ID. This can be a normal and existing RACF user ID without any additional RACF authorization attribute. The RACF command **CONNECT** may be used to connect this user ID with the RACF authorization attribute **SPECIAL** to the RACF group.

Each unit of work in the z/OS system that requires UNIX System Services must be associated with a valid UNIX System Services identity. A valid identity refers to the presence of a valid UNIX user ID (UID) and a valid UNIX group ID (GID) for each such user. The UID and the GID are defined through the OMVS segment in the user's RACF user profile and in the group's RACF group profile.

Each functional RACF access group must be authorized to access a specific TCP/IP RACF resource with a specific access attribute of either:

- ▶ ALTER
- ▶ CONTROL
- ▶ UPDATE
- ▶ READ
- ▶ NONE

This authorization is done with the RACF command **PERMIT** for each RACF resource needed by the functional access group.

Finally, RACF user IDs will be connected to the RACF resource access group using the RACF command **CONNECT**. The user ID then has the proper resource access authorization of the resource access group.

This example of a RACF implementation may seem confusing. However, it has some significant advantages over other RACF implementation approaches that are also possible. It isolates and shields RACF user IDs from the complex RACF resource definition and access authorization process. In addition it makes auditing and maintenance of the RACF environment easier. This approach to implementation is a question of organization and may be seen as follows:

- ▶ All RACF resources must be defined and have an owner assigned. An owner may be any RACF user ID or RACF group of the RACF cloud. If the owner is a group and this group is the focal point for all defined RACF resources of an entity, such as a decentralized structure, then its management is much simpler.
- ▶ RACF resources access authorization has to be given with specific access attributes to RACF user IDs and/or groups of the RACF cloud. If this RACF resource access authorization is given to functional resource access subgroups of the resource owning level, then its management, administration, and auditing is simplified. Being a resource access subgroup of the superior resource definition level means a strict isolation of both functional and hierarchical levels.
- ▶ RACF user IDs may have any number of personal resource access authorizations and be connected to any RACF group in the RACF cloud. The access rights may be the sum of all user IDs and connected groups' access group authorizations. If the user ID doesn't get any personal resource access authorization, then the process of management, administration and auditing is simplified again. It is easy to list and show all user ID connections to any authorization group. It is simple to list and show all access authorization given to a specific RACF resource access subgroup.

The RACF system-wide option **SETROPTS GRPLIST** must be set for the above RACF implementation to work.

Assigning user IDs to started tasks

Chapter 2, "Customizing UNIX System Services" on page 17 and *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)), both emphasize the need to associate user IDs and started tasks with an OMVS RACF segment. In some cases, the user ID and started task must be associated with the UNIX superuser. In other cases, you can associate the user ID and started task with the default user.

RACF offers you two techniques to assign user IDs and group IDs to started tasks:

1. The started procedure name table (ICHRIN03).
2. The RACF STARTED resource profiles.

By using the STARTED resources, you can add new started tasks to RACF, and immediately make those new definitions active.

```
IEF695I START T03DNS WITH JOBNAME T03DNS IS ASSIGNED TO USER TCPIP3
, GROUP OMVSGRP
```

The user ID and default group must be defined in RACF, which then treats the user ID as any other RACF user ID for its resource access checking. RACF allows multiple started procedure names to be assigned to the same RACF user ID. This has been used to assign two RACF user IDs to *all* TCP/IP started tasks as follows:

- ▶ User ID TCPIP3, to all started tasks requiring superuser authority
- ▶ User ID STCTCPIP3, to NFS (with the OPERATIONS attribute)

The default RACF group of STCTCPIP was assigned to STCTCPIP3. STCTCPIP has been defined as the RACF resource access subgroup of TCPOWN. This means that these RACF user IDs are implicitly connected to this RACF group.

The SUPERUSER RACF group of OMVSGRP was assigned to TCPIP3.

MVS VARY TCPIP commands

Access to MVS **VARY TCPIP** commands can be controlled by RACF by defining the commands (for example, VARY TCPIP OBEYFILE) to RACF class OPERCMDS. See *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)), for further information.

NETSTAT command

Access to the TSO **NETSTAT** command, the UNIX shell command **onetstat**, and command options can be controlled by RACF by defining NETSTAT resources to the RACF generic class SERVAUTH. For example, it is possible to prevent access to the **NETSTAT CONFIG** command, but allow access to **NETSTAT CONN**. See *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)) for further details.

More Information on RACF with CS for z/OS IP

RACF can be used to protect many TCP/IP resources, such as the TCP/IP stack itself and ports. Further information on securing your TCP/IP implementation can be found in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)).

3.4.3 TCP/IP server functions

Each CS for z/OS IP server relies on the use of an external security manager such as RACF. Several servers provide some built-in security functions for additional security. However, the following servers need some special attention:

- ▶ FTP
- ▶ NFS
- ▶ NCS
- ▶ RSHD

FTP security considerations

When the FTP server processes a logon request from a client user, it will issue a RACROUTE REQUEST=VERIFY, where the following parameters will be passed:

1. User's user ID
2. User's password and, optionally, a new password
3. The first seven characters of the FTP server jobname as application name
4. A terminal ID based on the IP address of the client

If you allow ANONYMOUS connections to your MVS FTP server, then all universally permitted data sets in your MVS system are available to all users in the TCP/IP network.

RACF considerations for FTP in CS for z/OS IP are covered in great detail in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

NFS security considerations

Network File System (NFS) requires that its started task user ID have the RACF attribute OPERATIONS. When NFS opens a data set on behalf of the network user, MVS OPEN uses the credentials of the NFS address space and not the credentials of the network user.

Please see *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228 for more information.

NCS security considerations

For the NCS administrator there is an administration module that allows the administrator to add, delete, or view entries in the location broker data sets. During installation of TCP/IP for MVS, this module is loaded into a separate load library: *tcpip.SEZALNK2*. Since there is no protection built into the module, there is only one way to prevent users from inadvertently updating the location broker data sets, and that is to restrict the use of this module to a few people responsible for administration of the contents of the location broker data sets. You may use RACF to protect the *tcpip.SEZALNK2* load library by giving it a universal access code of NONE and permitting only the NCS administrator(s) to read the data set.

Please see *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229 for more information.

RSHD server security considerations

If you allow your users to send remote shell (RSH) commands to your MVS remote execution server (REXECD) address space without an MVS password, you must enable the RACF surrogate job submission resource class, and add surrogate resource definitions to RACF.

Please see *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229 for details on this support.

3.4.4 TCP/IP client functions

The client functions of Communications Server for z/OS IP are executed in a TSO environment or a UNIX shell environment. Some functions are also available in other environments, such as batch or started task address spaces.

Any TSO user may execute any TCP/IP command and use a TCP/IP client function to access any other TCP/IP server host via the attached TCP/IP network. If these TCP/IP servers have not implemented adequate password protection, then any TSO client user may log on to these servers and access all data.

3.4.5 UNIX client functions

Certain client functions executed from the UNIX shell environment require superuser authority. The user ID accessing the shell must have an OMVS segment associated with it. You will find more about this subject in Chapter 2, "Customizing UNIX System Services" on page 17 and *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002)).

3.4.6 TCP/IP built-in security functions

TCP/IP for MVS has some built-in security functions that may be activated and used to control specific areas.

- ▶ The Simple Mail Transfer Protocol (SMTP) provides a secure mail gateway option that allows an installation to create a database of registered network job entry (NJE) users who are allowed to send mail through SMTP to a TCP/IP network recipient.
- ▶ The FTP server gives you the opportunity to code four security exits, in which you may extend the control over the functions performed by the FTP server. Using these exits you may control:
 - The use of the FTP server based on IP addresses and port numbers.
 - The use of the FTP server based on user IDs.
 - The use of individual FTP subcommands.
 - The submission of batch jobs via the FTP server.
- ▶ SNMP with Communications Server for z/OS IP has an SNMP agent that supports community-based security such as SNMPv1 and SNMPv2C, and user-based security such as SNMPv3. If you are concerned about sending SNMP data in a less secure environment, you may consider implementing SNMPv3, whose messages have data integrity and data origin authentication.

An SNMP subagent within CS for z/OS IP allows SET operations on some MIB objects. If you want to disable this function for security reasons, you may do so with parameters in the SACONFIG section of the PROFILE.TCPIP.

You may read more about the new SNMP in *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

- ▶ Both the IMS sockets and CICS sockets support provides a user exit that you can use to validate each IMS or CICS transaction received by the Listener function. How you code this exit and what data you require to be present in the transaction initiation request is up to you to decide. See *z/OS V1R2.0 CS: IP IMS Sockets Guide*, SC31-8830, and *z/OS V1R2.0 CS: IP CICS Sockets Guide*, SC31-8807, for full details.

3.5 Installation

The authoritative reference for any installation should be the *z/OS V1R2.0 Program Directory, Program Number 5694-A01*, GI10-0670. For this reason we do not include here a summary of this procedure. However, we emphasize to you the importance of working closely with the UNIX System Services and RACF implementers for your installation and customization of Communications Server for z/OS IP.

Having the Program Directory and understanding what it contains is not simply a matter of self-education and self-preservation. For proper teamwork with the z/OS installer, the UNIX System Services implementer, and the RACF administrator, you should know how your systems have been installed and prepared for you before you begin your customization. The implementer of CS for z/OS IP is encouraged to work closely with the z/OS installer, UNIX System Services implementer, and the RACF administrator.

Unless you have extensive experience with z/OS installation and maintenance procedures, much of the Program Directory can be a mystery to you. Do not let that discourage you. Examine the table of contents for the Program Directory and pick out the parts that affect your Communications Server for z/OS IP customization.

3.5.1 TCP/IP configuration data set names

This topic is described in *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775. We strongly recommend that you read the information about data set names in this book, before you decide on your data set naming conventions.

The purpose here is to give an introduction to the data set naming and allocation techniques used by CS for z/OS IP.

In early versions of TCP/IP for MVS, almost all configuration data sets were implicitly allocated. Starting with TCP/IP MVS Version 3, you had a choice for some of the configuration data sets, whether they should be allocated implicitly or explicitly. With CS for z/OS IP, you still have a choice in the matter, but in addition you need to ensure that not only MVS functions find the appropriate data sets, but also that the UNIX System Services functions do as well.

- ▶ **Implicit**

The name of the configuration data set is resolved at runtime based on a set of rules (the search order) implemented in the various components of TCP/IP. When a data set name has been resolved, the TCP/IP component uses the dynamic allocation services of MVS and/or of UNIX System Services to allocate that configuration data set. The various data sets, whether for MVS functions or for UNIX System Services functions, are described in *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775, and the chapter that describes a specific server or client process. The information we provide below should be considered merely introductory, since you must examine the documentation on each procedure you use to determine what its specific resolution method is.

These are some of the data sets (or files) that can only be implicitly allocated in CS for z/OS IP:

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

In the above data set names, hlq is determined using the following search sequence:

- a. User ID or jobname
- b. DATASETPREFIX value (or its default of TCPIP), defined in TCPIP.DATA.

Dynamically allocated data sets can include a mid-level qualifier (MLQ), for example, a node name, or a function name:

- a. For data sets containing a PROFILE.TCPIP configuration file:
 - xxxx.nodename.zzzz
- b. For data sets containing a translate table used by a particular TCP/IP server:
 - xxxx.function_name.zzzz (for the FTP server the function_name is SRVRFTP)

Data set SYS1.TCPPARMS(TCPDATA) can be dynamically allocated if it contains the TCPIP.DATA config file.

► **Explicit**

For some of the configuration files, you can tell TCP/IP which files to use by coding DD statements in JCL procedures, or by setting UNIX environment variables. If the required data set has been allocated, for example, via a DD statement in the JCL used to start a TCP/IP component, the TCP/IP component will read its configuration data from that allocation, and will not try to construct a configuration data set name for dynamic allocation. Again, the various data sets used by TCP/IP functions and their resolution method are described in *z/OS V1R2.0 CS: IP Configuration Guide, SC31-8775* in the chapter that describes a specific server or client process.

3.5.2 Installation steps

Your z/OS installer must follow the z/OS installation steps detailed in the *z/OS V1R2.0 Program Directory, Program Number 5694-A01, GI10-0670*. The following list includes only a high-level view of the steps required, in order to give you, the CS for z/OS IP implementer, a feeling of the whole installation process. The steps, unless otherwise indicated, are addressed to the z/OS installer.

1. Clone the running z/OS system. This becomes the target system. Verify that the clone IPLs.
2. Back up your cloned system.
3. Update the SMP/E entries for APPLY processing.
4. Decide which FMIDs to install.
 - For Communications Server for z/OS IP, you must have:
 - DFSMS since this is a prerequisite for an HFS system
 - RACF, another prerequisite for CS for z/OS IP
 - VTAM, a prerequisite for the data link control of CS for z/OS IP
5. Install products in *waves*.

Waves are further broken down into subsections called *ripples*. The individual products that are installed in each wave and ripple are documented in the *z/OS V1R2.0 Program Directory, Program Number 5694-A01*.

- Wave 0 installs those elements that should be available on the driving system for subsequent wave installs.
- Wave 1 installs all FMIDs that don't install into an HFS.
 - Some of CS for z/OS IP installation occurs with this wave, including the necessary DFSMS, RACF, and VTAM.
- Wave 2 installs those elements that install into an HFS.
 - Some of CS for z/OS IP installation occurs with this wave, including the HFS portion of TCP/IP.

The z/OS V1R2.0 Program Directory, Program Number 5694-A01 points the CS for z/OS IP implementer to *z/OS V1R2.0 CS: IP Migration, GC31-8773*, for completing some of these steps.

- Run the EXEC to create the HFS directory systems and paths during this wave (EZAQEMDR).
- Customize the PARMLIB members, including BPXPRMxx for UNIX System Services.
- Customize BPXPRMxx for CS for z/OS IP now or wait until wave 3 completes.
- The CS for z/OS IP implementer may build the TCP/IP profile and Resolver configuration data sets or files at this point or may wait until wave 3 completes.

The z/OS V1R2.0 Program Directory, Program Number 5694-A01 points the TCP/IP implementer to *z/OS V1R2.0 CS: IP Migration, GC31-8773*, for completing some of these steps.

- Customize the RACF definitions to perform TCP/IP work or perform this after wave 3 completes.

- Wave 3 installs the z/OS level of the JES2 or JES3 elements.

6. Back up the system after the successful APPLY step for each wave.
7. Back up the system after the successful ACCEPT step for each wave.
8. Back up the system after each successful IPL of the system.

MVS system modifications will have occurred during the above installation process. If you, the CS for z/OS IP system programmer, used to perform these tasks in the past, you may be surprised to find that they have already been performed by the z/OS system programmer. You will probably want to verify what was done during this process, so we suggest that you review the list of modifications made to your system. In “Updating the MVS system data sets” on page 61 and “Other modifications to the MVS system” on page 62, you find a list of items you were accustomed to finding in the formerly separate TCP/IP program directories.

Updating the MVS system data sets

z/OS uses the concept of *product enablement policy*. Certain products require registration in SYS1.PARMLIB(IFAPRDxx) if you intend to use them. If you forget to enable Communications Server for z/OS IP, you will see a message like the following:

```
IFA104I REGISTRATION HAS BEEN DENIED FOR PRODUCT WITH OWNER='IBM CORP'  
      NAME=z/OS FEATURE=featurename VERSION=vv.rr.mm ID=5694-A01
```

Message IFA104I indicates that the product has not been listed in IFAPRDxx:

```
IFA104I REGISTRATION HAS BEEN DENIED FOR PRODUCT WITH OWNER=prodown
      NAME=prodname FEATURE=featurename VERSION=vv.rr.mm ID=prodid
```

Explanation: The system denied the product's request to register.

The product has a state of DISABLED in the product enablement policy. The product is not defined in the policy but its register request indicated that it should be disabled when there is no entry in the policy.

System Programmer Response: ...

(Check the enablement policy.)

You might need to change the product's state from DISABLED to ENABLED.

To correct the problem, just add the following entry to IFAPRDxx:

```
PRODUCT OWNER('IBM CORP')
      NAME(z/OS)
      ID(5694-A01)
      VERSION(*) RELEASE(*) MOD(*)
      FEATURENAME('TCP/IP BASE')
      STATE(ENABLED)
```

Other modifications to the MVS system

Updating the MVS system libraries must be done with great care. Please follow the instructions in the *z/OS V1R2.0 Program Directory, Program Number 5694-A01, GI10-0670*, any additional hints in the PSP bucket, and information in *z/OS V1R2.0 CS: IP Migration, GC31-8773*, to ensure that all required MVS system modifications are done as required. You may need to make changes to all of the following members, depending on the features you are installing.

- ▶ SYSx.PARMLIB(PROGxx) or SYSx.PARMLIB(IEAAPFxx)
To APF-authorize CS for z/OS IP load libraries.
- ▶ SYSx.PARMLIB(LNKLSTxx)
To add CS for z/OS IP link libraries to the MVS system link list.
- ▶ SYSx.PARMLIB(LPALSTxx)
To add the CS for z/OS IP LPA modules to the LPA during IPL of MVS.
- ▶ SYSx.PARMLIB(IEFSSNxx)
To specify the VMCF and TNF subsystem names and optionally initialize the subsystems.
- ▶ SYSx.PARMLIB(SCHEDxx)
To specify certain CS for z/OS IP modules as privileged modules in MVS.
- ▶ SYSx.PARMLIB(IKJTSoxx)
To specify CS for z/OS IP modules as authorized TSO commands.
- ▶ SYSx.PARMLIB(IFAPRDxx) or (PROGxx)
To add product and feature information in a z/OS environment.
- ▶ SYS1.PROCLIB or the installation PROCLIB you use for your TCP/IP-related procedures.
For all your TCP/IP JCL procedures.

Please use the following checklist as an introduction to the process of modifying the MVS system:

1. SYSx.PARMLIB updates:

a. LNKSTxx

Add the following data sets:

hlq.SEZALINK

hlq.SEZALNK2

b. LPALSTxx

Add the following data set:

hlq.SEZALPA

Note: hlq.SEZALPA must be cataloged into the MVS master catalog. hlq.SEZALINK and hlq.SEZALNK2 can be cataloged into the MVS master catalog. You may omit them from the MVS master catalog if you identify them to include a volume specification as in:

```
TCPIP.SEZALINK(WLTCP),  
TCPIP.SEZALNK2(WLTCP)
```

If the three data sets mentioned were renamed during the installation process, then use these names instead.

c. PROGnn or IEAAPFxx

Add the following TCP/IP libraries for APF authorization:

hlq.SEZATCP

hlq.SEZADSIL

hlq.SEZALINK

hlq.SEZALNK2

hlq.SEZALPA

hlq.SEZAMIG

d. IEFSSNxx

TNF and VMCF are required for CS for z/OS IP. Add the subsystem definitions for the MVS address spaces of TNF and VMCF as follows:

- If you choose to use restartable VMCF and TNF:

TNF

VMCF

- If you will not be using restartable VMCF and TNF:

TNF,MVPTSSI

VMCF,MVPXSSI,*nodename*

The *nodename* should be set to the MVS NJE node name of this MVS system. It is defined in the JES2 parameter member of SYSx.PARMLIB:

```
NJEDEF    ....
          OWNNODE=03,
          ....
```

```
N03      NAME=SA03,SNA,NETAUTH
```

Make sure that the hlq.SEZALINK definition has been added to LNKLSTxx and the library itself has been APF authorized before you make this update. MVS initializes the address spaces of the TNF and VMCF subsystems during IPL as part of the master scheduler initialization.

e. SCHEDxx

The entries below are present in the IBM-supplied program properties table (PPT). However, if your installation has a customized version of the PPT, ensure these entries are present:

- For CS for z/OS IP

```
PPT PGMNAME(EZBTCPIP) KEY(6) NOCANCEL PRIV NOSWAP SYST LPREF SPREF
```

- If you use restartable VMCF and TNF

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

- For NPF

```
PPT PGMNAME(EZAPPFS) KEY(1) NOSWAP
PPT PGMNAME(EZAPAAA) NOSWAP
```

- For SNALINK

```
PPT PGMNAME(SNALINK) KEY(6) NOSWAP SYST
```

f. COMMNDxx

VMCF and TNF are required for CS for z/OS IP. If you use restartable VMCF and TNF, procedure EZAZSSI must be run during your IPL sequence (EZAZSSI starts VMCF and TNF). Either use your operation's automation software to start EZAZSSI, or add a command to your COMMNDxx member in SYSx.PARMLIB:

```
COM='S EZAZSSI,P=your_node_name'
```

The value of variable P defaults to the value of the MVS symbolic &SYSNAME. If your node name is the same as the value of &SYSNAME, then you can use the following command instead:

```
COM='S EZAZSSI'
```

When the EZAZSSI address space starts, a series of messages is written to the MVS log indicating the status of VMCF and TNF; then the EZAZSSI address space terminates. Once VMCF and TNF have initialized successfully, you can start your TCP/IP system address spaces.

g. IKJTSOxx

Update the IKJTSOxx member by adding the following to the AUTHCMD section: MVPXDISP, NETSTAT, TRACERTE, RSH, LPQ, LPR and LPRM.

h. IEASYSxx

Review your CSA and SQA specifications and verify that the numbers allocated are sufficiently large enough to prevent `getmain` errors.

```
IEASYSxx: CSA(3000,250M) (used in BETA accounts)
IEASYSxx: SQA(8,448)    (used in BETA accounts)
```

i. IVTPRMxx

Review the computed CSM requirements to reflect z/OS VTAM and CS for z/OS IP usage:

- IVTPRMxx: FIXED MAX(120M)
- IVTPRMxx: ECSA MAX(30M)

j. CTIEZBxx

Copy the following member to SYSx.PARMLIB from hlq.SEZAINST for use with CTRACE:

CTIEZB00 can be customized to include a different sized buffer. The default buffer size is 8MB. We made a new member, CTIEZB01, with the buffer size change. You may read more about the use of component tracing (CTRACE) in *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782 and in *z/OS V1R2.0 CS: IP Migration*, GC31-8773.

k. BPXPRMxx

- Define CS for z/OS IP as a UNIX Physical File System. At a minimum you must have the following INET definition. However, if you are using CINET, then you will need to make adjustments as described in Chapter 2, “Customizing UNIX System Services” on page 17 and Chapter 5, “Multiple TCP/IP stacks on z/OS” on page 117.

```
FILESYSTYPE  TYPE(INET) ENTRYPOINT(EZBPFINI)
NETWORK      DOMAINNAME(AF_INET)
              DOMAINNUMBER(2)
              MAXSOCKETS(60000)
              TYPE(INET)
              INADDRANYPORT(4000)
              INADDRANYCOUNT(2000)
```

Ensure that the INADDRANYPORT assignment does not conflict with PORT assignments in the PROFILE.TPCIP data set.

Note: The OpenEdition ENTRYPOINT for TCP/IP for MVS OpenEdition Applications Feature was BPXTIINT; for CS for z/OS IP it is EZBPFINI. If you have failed to make this change in an existing BPXPRMxx member, you may see messages such as EZZ4203I or abend codes such as S806.

- Review the values specified in BPXPRMxx for MAXPROCSYS, MAXPROCUSER, MAXUIDS, MAXFILEPROC, MAXPTYS, MAXTHREADTASKS, and MAXTHREADS.
- Update BPXPRMxx in SYS1.PARMLIB with the following:

```
MOUNT FILESYSTEM('0EA.TCPIP.HFS')
      TYPE(HFS)
      MODE(RDWR)
      MOUNTPOINT('usr/lpp/tcpip')
```

2. SYS1.PROCLIB or the PROCLIB you use for your TCP/IP JCL procedures.

- a. If you choose to use restartable VMCF and TNF, add procedure EZAZSSI:

```
//EZAZSSI PROC P=' '
//STARTVT EXEC PGM=EZAZSSI,PARM=&P
//STEPLIB DD DSN=hlq.SEZATCP,DISP=SHR
```

- b. Update your TCP/IP startup JCL procedure. The sample for the CS for z/OS IP procedure is in hlq.SEZAINST(TCPIPPROC).

3. The PROCLIB you use for your TSO logon procedures.

Update your TSO logon procedures by adding the TCP/IP help data set hlq.SEZAHHELP to the //SYSHLP DD concatenation. Optionally, add the //SYSTCPD DD statement to your logon procedures.

Add hlq.SEZAMENU to the //ISPLIB DD concatenation and hlq.SEZAPENU to the //ISPLIB DD and the //ISPTLIB DD concatenations.

4. Establish RACF security environment for Communications Server for z/OS IP.

The notes that follow are merely an overview. You or the RACF implementer should consult the instructions in *Secure e-business in TCP/IP Networks on OS/390 and z/OS*, SG24-5383, and *z/OS V1R2.0 CS: IP Migration*, GC31-8773, to accomplish these tasks.

a. Defining commands for CS for z/OS IP in the RACF OPERCMDS class

b. Establishing a Group ID for a default OMVS group segment

```
ADDGROUP OEDFLTG OMVS(GID(9999))
```

c. Defining a user ID for a default OMVS group segment

```
RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('OEDFLTU/OEDFLTG')
ADDUSER OEDFLTU DFLTGRP(OEDFLTG) NAME('OE DEFAULT USER') PASSWORD(xg18ej)
OMVS(UID(999999) HOME('/') PROGRAM('/bin/sh'))
```

d. Activating or refreshing appropriate facility classes

```
SETOPTS CLASSACT(FACILITY)
SETOPTS RACLIST(FACILITY)
SETOPTS RACLIST(FACILITY) REFRESH
```

e. Defining one or more superuser IDs to be associated with certain UNIX System Services users and TCP/IP started tasks

```
ADDGROUP OMVSGRP OMVS(GID(1))
ADDUSER TCPIP3 DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))
```

f. Defining other UNIX System Services users

You may already have defined RACF groups and users. If this is the case, you may merely need to set up an HFS home directory for each user, add an OMVS identity by altering the group to include a GID (ALTGROUP), and then using the ISHELL utility to add OE segments for UNIX System Services users, associating them with the altered group and giving each user a distinct UID.

Otherwise you may have to perform these tasks in a more painstaking manner:

```
ADDGROUP usergrp OMVS(GID(10))
ADDUSER user01 DFLTGRP(usergrp) OMVS(UID(20) HOME('/u/user01') PROGRAM('/bin/sh'))
```

3.6 Message types: Where to find them

You will want to have the z/OS System Messages manuals available as well as the messages manuals for TCP/IP. You will want to understand messages with a BPX prefix, an EZn prefix, SNA sense codes, and DLC codes.

3.6.1 Messages with prefix of BPX

You will find the explanations for these messages in *z/OS V1R2.0 MVS System Messages, Vol 3 (ASB-BPX)*, SA22-7633.

3.6.2 Messages with prefix of EZA

For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 1 (EZA)*, SC31-8783.

3.6.3 Messages with prefix of EZB

For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 2 (EZB)*, SC31-8784.

3.6.4 Messages with prefix of EZY

For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 3 (EZY)*, SC31-8785.

3.6.5 Messages with prefix of EZZ

For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 4 (EZZ-SNM)*, SC31-8786.

3.6.6 Messages with prefix of FOM and FSUM

You will find the explanations for these messages in *z/OS V1R2.0 UNIX System Services Messages and Codes*, SA22-7807.

3.6.7 Eight-digit SNA sense codes and DLC codes

You will find the explanations for these codes in *z/OS V1R2.0 CS: IP and SNA Codes*, SC31-8791.

3.7 Checklist for installation and customization

1. Have you access to the detailed migration plan, fallback plan, and test plan for your installation?
2. Have you obtained at a minimum the following documentation:
 - *z/OS V1R2.0 Program Directory, Program Number 5694-A01*, GI10-0670
 - Preventive Service Planning (PSP) bucket
 - Upgrade name = OS390Rn; where *n* is 5, 6 or 7 for the release and subset ID = CS390IP
 - *z/OS V1R2.0 UNIX System Services Planning*, SA22-7800
 - *z/OS V1R2.0 UNIX System Services User's Guide*, SA22-7801
 - *z/OS V1R2.0 UNIX System Services Messages and Codes*, SA22-7807
 - *z/OS V1R2.0 UNIX System Services Command Reference*, SA22-7802
 - *z/OS V1R2.0 MVS System Messages, Vol 1 (ABA-AOM)*, SA22-7631
 - *z/OS V1R2.0 MVS System Messages, Vol 2 (ARC-ASA)*, SA22-7632
 - *z/OS V1R2.0 MVS System Messages, Vol 3 (ASB-BPX)*, SA22-7633
 - *z/OS V1R2.0 MVS System Messages, Vol 4 (CBD-DMO)*, SA22-7634

- *z/OS V1R2.0 MVS System Messages, Vol 5 (EDG-GFS), SA22-7635*
 - *z/OS V1R2.0 MVS System Messages, Vol 6 (GOS-IEA), SA22-7636*
 - *z/OS V1R2.0 MVS System Messages, Vol 7 (IEB-IEE), SA22-7637*
 - *z/OS V1R2.0 MVS System Messages, Vol 8 (IEF-IGD), SA22-7638*
 - *z/OS V1R2.0 MVS System Messages, Vol 9 (IGF-IWM), SA22-7639*
 - *z/OS V1R2.0 MVS System Messages, Vol 10 (IXC-IZP), SA22-7640*
 - *z/OS V1R2.0 CS: IP Migration, GC31-8773*
 - *z/OS V1R2.0 CS: IP Configuration Guide, SC31-8775*
 - *z/OS V1R2.0 CS: IP Configuration Reference, SC31-8776*
 - *z/OS V1R2.0 CS: IP User's Guide and Commands, SC31-8780*
 - *z/OS V1R2.0 CS: IP Messages Volume 1 (EZA), SC31-8783*
 - *z/OS V1R2.0 CS: IP Messages Volume 2 (EZB), SC31-8784*
 - *z/OS V1R2.0 CS: IP Messages Volume 3 (EZY), SC31-8785*
 - *z/OS V1R2.0 CS: IP Messages Volume 4 (EZZ-SNM), SC31-8786*
 - *z/OS V1R2.0 CS: IP and SNA Codes, SC31-8791*
 - Any other CS for z/OS IP manuals suitable for your product set
3. Have TNF and VMCF initialized successfully?
Check console log for a successful start of EZAZSSI, TNF, VMCF.
 4. Has the TCP/IP feature of z/OS been enabled or registered in IFAPRDxx?
 5. Has a *full-function* OMVS (DFSMS, RACF, HFS) started successfully?
 - a. Is OMVS active when you issue `D OMVS`?
 - b. Is SMS active when you issue `D SMS`?
 - c. Have HFS file systems been mounted? Verify with `D OMVS, F`.
 - d. Is RACF enabled on the system?
 6. Have the definitions in BPXPRMxx of SYS1.PARMLIB been made to reflect:
 - a. The correct transport provider for the stack(s) you will be running?
 - b. The correct CS for z/OS IP proc names?
 - c. The correct use of INET versus CINET?
 - d. The correct ENTRYPOINT name for Communications Server for z/OS IP versus earlier versions of OE function in TCP/IP? (z/OS IP ENTRYPOINT = EZBPFINI)
 - e. The mounting of filesystems for users? (You can verify with `D OMVS, F`.)
 - f. Appropriate values for MAXPROCSYS, MAXPROCUSER, MAXUIDS, MAXFILEPROC, MAXPTYs, MAXTHREADTASKS, and MAXTHREADS.
 7. Have HFS file systems and directories been created and mounted for the users of the system?
 8. Have RACF definitions been put in place for:
 - a. OMVS user IDs and group IDs for your CS for z/OS IP procedures
 - b. OMVS user IDs and group IDs for your other users, for Superusers, for a Default User, with definitions for appropriate Facility Classes, like BPX.SUPERUSER
 - c. TCP/IP **VARY** commands

d. NETSTAT commands

9. Have you placed the correct definitions in the MVS data sets:

- LNKLSTxx
- LPALSTxx
- SCHEDxx
- PROGxx
- IEASYSxx
- IEFSSNxx
- IKJTSOxx
- IVTPRMxx

10. Do you know how to work your way around the ISHELL and/or issue UNIX commands from the shell environment? (see *z/OS V1R2.0 UNIX System Services Command Reference*, SA22-7802.)

Have you attended a class on working with the shell environment or have you obtained hands-on experience otherwise?

11. Do you know how to set the environment variables for UNIX procedures so that the correct Resolver configuration data sets are found?

12. Raw sockets require authorization; they run from SEZALINK and are usually already authorized; if you have moved applications and functions to another library (not recommended), ensure that this library is authorized.

13. The loopback address is now 127.0.0.1. If you require 14.0.0.0, have you added this to the HOME list?

14. Have you computed CSA requirements to include not only z/OS VTAM, but also CS for z/OS IP?

- a. IEASYSxx: CSA(3000,250M) (used in BETA accounts)
- b. IEASYSxx: SQA(8,448) (used in BETA accounts)

15. Have you computed CSM requirements to include not only z/OS VTAM, but also CS for z/OS IP?

- a. IVTPRMxx: FIXED MAX(120M)
- b. IVTPRMxx: ECSA MAX(30M)

16. Have you modified the CTRACE initialization member to reflect at least 4 MB of buffer storage? (CTIEZB00)

17. Have you created CTRACE Writer procedures for taking traces?

18. Have you updated your TCP/IP Proc?

19. Have you updated your other procs, for example, the FTP server proc?

20. Have you revamped your TCP/IP Profile to use the new statements and to comment out the old?

- a. Have you made provisions to address 3172 connections that are no longer supported?
- b. Have you investigated all your connections to ensure to what extent they are still supported? (In some cases, definitions will have changed.)

21. Have your applications that relied on VMCF and IUCV sockets been converted now that those APIs are no longer supported?

22. If you plan to use multiple Communications Server for z/OS IP stacks, have you familiarized yourself with the issues surrounding PORT allocations? (See Chapter 5, "Multiple TCP/IP stacks on z/OS" on page 117.)

23. Have you reviewed the Planning and Migration checklist in Appendix B of *z/OS V1R2.0 CS: IP Migration*, GC31-8773 and made appropriate plans to use the sample data sets?
24. Have you reviewed the list and location of configuration data set samples in Chapter 1 of *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776?



Configuring base functions

This chapter describes how to customize the basic functions of the TCP/IP system address space. We will also review some of the commands to display the configuration and status of CS for z/OS IP resources. These commands may be issued from TSO, from the UNIX System Services shell environment, or as MVS console commands.

Starting with OS/390 V2R5 IP, the OS/390 TCP/IP stack is now fully integrated with UNIX System Services and capable of supporting many traditional TCP/IP applications and UNIX applications. This requires configuration steps that must be coordinated between the implementer of both CS for z/OS IP and UNIX System Services.

The installation of Communications Server for z/OS IP is detailed in the *z/OS V1R2.0 Program Directory, Program Number 5694-A01, GI10-0670*, Chapter 2, “Customizing UNIX System Services” on page 17, and Chapter 3, “Installation” on page 49. The customization of UNIX System Services and Communications Server for z/OS IP is described in *z/OS V1R2.0 UNIX System Services Planning, GA22-7800*.

You have already read that several jobs must be accomplished to arrive at a successfully operating Communications Server for z/OS IP:

- ▶ Implement a *full-function* UNIX System Services system on z/OS.
- ▶ An overview of a *full-function* UNIX System Services implementation is described in Chapter 1, “Communications Server for z/OS IP overview” on page 1 and Chapter 2, “Customizing UNIX System Services” on page 17.
- ▶ Define a RACF user ID with an OMVS UID and assign it to the started task name of the CS for z/OS IP system address space that is going to be used as your z/OS UNIX AF_INET transport provider.
- ▶ Information on RACF user IDs for UNIX System Services and Communications Server for z/OS IP is documented in Chapter 2, “Customizing UNIX System Services” on page 17 and *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security, SG24-6840*.
- ▶ Customize SYS1.PARMLIB(BPXPRMxx) to use either the integrated sockets AF_INET physical file system, or the Common AF_INET physical file system.

This process is described in Chapter 2, “Customizing UNIX System Services” on page 17 and Chapter 5, “Multiple TCP/IP stacks on z/OS” on page 117.

- ▶ Customize your CS for z/OS IP configuration data sets:
 - PROFILE.TCPIP
 - TCPIP.DATA
 - Other configuration data sets

Samples of the configuration files can be found in hlq.SEZAINST, and in the HFS directory /usr/lpp/tcpip/samples.

4.1 z/OS IP Configuration Wizard and msys for Setup

The z/OS IP Wizard and msys for Setup are IBM offerings that can help you configure your CS for z/OS IP system. Both can be used to generate CS for z/OS IP configuration files.

4.1.1 z/OS IP Configuration Wizard

The z/OS IP Configuration Wizard sets up basic IP configuration for a single stack, including simple instances of OMPROUTE, FTP and TN3270 servers.

The wizard takes you through a series of screens where you enter information about your z/OS IP configuration: general information such as the host name, the domain name, and the address of a name server; information about the type of routing you will use, details about your network connections, and whether you want to configure FTP and TN3270 servers. Wherever possible, the wizard uses defaults.

Figure 4-1 Example of Wizard screen

The wizard then builds customized PROFILE.TCPIP, TCPIP.DATA and OMPROUTE.CONF files, based on the information entered on the screens. You can upload the customized files to your system. The wizard also provides a checklist of tasks that must be completed when configuring z/OS IP.

The wizard can be found at:

<http://www.ibm.com/eserver/zseries/zos/wizards>

4.1.2 z/OS msys for Setup

A z/OS system is very complex. It is controlled using a large variety of settings, including parmlib members, /etc files for UNIX System Services and so on, each of which has a different interface, and requires special knowledge to configure. Managed System Infrastructure for Setup (msys for Setup) addresses these issues by establishing a central directory for product configuration data and a single interface to that directory.

msys for Setup code runs on a z/OS system. Product configuration data is held in a directory on an LDAP server running on z/OS. msys parses existing z/OS configuration files and stores the information in the LDAP directory.

The configuration data stored in the LDAP directory can be viewed and updated, and new configuration data added, via the windows of the msys Windows NT application. To enable this, the Windows NT workstation has IP connections to the LDAP server and the z/OS system. Many products, such as CS for z/OS IP, can be configured using the same msys application.

The current version of msys for Setup allows you to set up a basic TCP/IP system. The windows prompt you for information about your z/OS IP system, which is then stored in the LDAP directory. msys for Setup can then be used to generate TCP/IP configuration files PROFILE.TCPIP, TCPIP.DATA and OMPROUTE.CONF based on the information in the LDAP directory.

For further information on msys for Setup, see *z/OS V1R2.0 msys for Setup User's Guide*, SC33-7985.

4.2 PROFILE.TCPIP

Before you start your TCP/IP system, you must configure the operational and address space characteristics of your CS for z/OS IP stack.

These definitions are entered into a PROFILE configuration data set that is read by the TCP/IP system address space during initialization.

A sample PROFILE.TCPIP config file is provided in hlq.SEZAINST(SAMPPROF).

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

- ▶ TCP/IP operating characteristics
- ▶ TCP/IP reserved port number definitions
- ▶ TCP/IP physical network and hardware definitions
- ▶ TCP/IP Telnet definitions
- ▶ TCP/IP network routing definitions

In this chapter, we discuss TCP/IP operating characteristics and port reservation.

For information on TCP/IP physical network and hardware definitions, see *Communications Server for z/OS V1R2 Implementation Guide Volume 4 : Connectivity and Routing*, SG24-6516.

For information on TCP/IP Telnet definitions, see Chapter 8, "TN3270 Telnet server" on page 175.

For information on TCP/IP network routing definitions, see *Communications Server for z/OS V1R2 Implementation Guide Volume 4 : Connectivity and Routing*, SG24-6516.

4.2.1 Displaying the TCP/IP Config

To display the operating characteristics of a TCP/IP stack, enter any of the following commands:

- ▶ TSO/E command **NETSTAT CONFIG**
- ▶ MVS command **D TCPIP,procname,NETSTAT,CONFIG**

► UNIX shell command **onetstat -f**

The following is example output from the NETSTAT CONFIG display:

```
D TCPIP,TCPIPA,N,CONFIG

EZZ2500I NETSTAT CS V1R2 TCPIPA 372

TCP CONFIGURATION TABLE: 1
DEFAULTRCVBUFSIZE: 00016384  DEFAULTSNDBUFSIZE: 00016384
DEFLTMAXRCVBUFSIZE: 00262144
MAXRETRANSMITTIME: 120.000  MINRETRANSMITTIME: 0.500
ROUNDTRIPGAIN: 0.125  VARIANCEGAIN: 0.250
VARIANCEMULTIPLIER: 2.000  MAXSEGLIFETIME: 60.000
DEFAULTKEEPALIVE: 00000120  LOGPROTOERR: 00
RESTRICTLOWPORT: YES  SENDGARBAGE: NO
TCPTIMESTAMP: YES  FINWAIT2TIME: 600

UDP CONFIGURATION TABLE: 2
DEFAULTRCVBUFSIZE: 00065535  DEFAULTSNDBUFSIZE: 00065535
CHECKSUM: 00000001  LOGPROTOERR: 01
RESTRICTLOWPORT: YES  NOUDPQUEUELIMIT: NO

IP CONFIGURATION TABLE: 3
FORWARDING: NO  TIMETOLIVE: 00064  RSMTIMEOUT: 00060
FIREWALL: 00000
ARPTIMEOUT: 01200  MAXRSMSSIZE: 65535
IGREDIRECT: 00000  SYSPLXROUT: 00000  DOUBLENOP: 00000
STOPCLAWER: 00000  SOURCEVIPA: 00001  VARSUBNET: 00000
MULTIPATH: NO  PATHMTUDSC: 00000  DEVTRYDUR: 000000090
DYNAMICXCF: 00000
IQDIOROUTE: NO

SMF PARAMETERS: 4
TYPE 118:
  TCPINIT: 00  TCPTERM: 00  FTPCLIENT: 00
  TN3270CLIENT: 00  TCPIPSTATS: 00
TYPE 119:
  TCPINIT: NO  TCPTERM: NO  FTPCLIENT: NO
  TCPIPSTATS: NO  IFSTATS: NO  PORTSTATS: NO
  STACK: NO  UDPTERM: NO  TN3270CLIENT: NO

GLOBAL CONFIGURATION INFORMATION: 5
TCPIPSTATS: 00  ECSALIMIT: 0000000K  POOLLIMIT: 0000000K
```

Parameters such as SOURCEVIPA can be either ENABLED or DISABLED. A value of 01 in the NETSTAT CONFIG display means it is ENABLED.

1 shows what settings are in effect in the TCPCONFIG parameters.

2 shows what you have set for the UDPCONFIG parameters.

3 shows the settings in effect in the IPCONFIG parameters.

4 shows what is in effect for SMFCONFIG.

5 shows the settings in effect for GLOBALCONFIG.

4.2.2 Locating PROFILE.TCPIP

The following search order is used to locate the PROFILE.TCPIP config file:

1. //PROFILE DD DSN=.... (Explicit Allocation)
 - //PROFILE DD DSN=TCP.TCPPARMS(PROF03A)
2. jobname.nodename.TCPIP (Implicit Allocation)
3. hlq.nodename.TCPIP (Implicit Allocation)
4. jobname.PROFILE.TCPIP (Implicit Allocation)
5. hlq.PROFILE.TCPIP (Implicit Allocation)

The PROFILE must exist. Otherwise, the TCP/IP address space will terminate abnormally with the message:

```
EZZ0332I DD:PROFILE NOT FOUND. CONTINUING PROFILE SEARCH  
EZZ0325I INITIAL PROFILE COULD NOT BE FOUND
```

We recommend that you use //PROFILE DD statement in the TCP/IP system address space JCL procedure to explicitly allocate the PROFILE data set.

4.2.3 Configuration features of CS for z/OS IP

There have been a lot of new features and enhancements since OS/390 V2R5 IP. The following lists some of the more popular functions you might want to consider in configuring your TCP/IP stack:

- ▶ Multipath
- ▶ Path MTU discovery
- ▶ Dynamic XCF
- ▶ OSA-express Gigabit Ethernet support through the MPCIPA interface
- ▶ HiperSockets
- ▶ Sysplex Distributor
- ▶ Enterprise Extender

However, if you are migrating from a level of TCP/IP earlier than OS/390 V2R5 IP, you need to take note of the following additional changes:

1. Device support for MPC, native ATM, XCF and SAMEHOST interface.
 - MPC and ATM need VTAM TRL definitions (XCF and SAMEHOST TRLs are dynamically generated by VTAM).
2. Changed device support: specification of AUTORESTART and IFSPEED, IFHSPEED.
3. Dropped support for obsolete devices: DDN1822, IUCV, CETI, X25ICA, HIPPI, Offload for 3172.
4. Use of IPCONFIG ARPTO, specified in seconds, versus ARPAGE, specified in minutes.
5. Changed PORT reservation statements and PORTRANGE:
 - SHAREPORT (sharing a port between multiple listeners, TCP only)
 - BIND (making an application bind to a specific IP address)
6. The DELETE statement allows the removal of ATMARPSV, ATMLIS, ATMPVC, DEVICE, LINK, PORT and PORTRANGE definitions without restarting the TCP/IP stack.
7. New groupings for stack characteristics: parameters that were defined on the ASSORTEDPARMS statement should be migrated to the IPCONFIG, TCPCONFIG, UDPCONFIG and GLOBALCONFIG statements.

8. ITRACE and PKTTRACE replace TRACE, LESSTRACE, MORETRACE, NOTRACE.
9. Use of SMFCONFIG statement to turn on SMF logging.
10. Use of SACONFIG statement to configure the SNMP subagent.
11. New and changed miscellaneous statements:
 - TRANSLATE, INCLUDE, GLOBALCONFIG

The following parameters are obsolete:

- ▶ ADDRESSTRANSLATIONPOOLSIZE
- ▶ CCBPOOLSIZE
- ▶ ENVELOPEPOOLSIZE
- ▶ IPROUTEPOOLSIZE
- ▶ HPNSSTAGINGBUFFERS
- ▶ LARGEENVELOPEPOOLSIZE
- ▶ RCBPOOLSIZE
- ▶ SCBPOOLSIZE
- ▶ SKCPOOLSIZE
- ▶ SMALLDATABUFFERPOOLSIZE
- ▶ TCPPOOLSIZE
- ▶ TINYDATABUFFERPOOLSIZE
- ▶ UCBPOOLSIZE
- ▶ INFORM
- ▶ OFFLOADAPIOTHER
- ▶ RESTRICT
- ▶ SCREEN, NOSCREEN
- ▶ SYSCONTACT, SYSLOCATION
- ▶ TIMESTAMP
- ▶ OBEYLIST

For further information on migrating to Communications Server for z/OS IP, refer to *z/OS V1R2.0 CS: IP Migration*, GC31-8773.

4.2.4 System symbolics

Prior to OS/390 V2R7 IP, each TCP/IP profile configuration data set is unique. This means that if you are running your TCP/IP stacks in a sysplex, you would need to maintain one configuration for each stack on each of the systems. As more systems are added to the sysplex, more TCP/IP configuration files need to be maintained and synchronized.

Here in the ITSO we used system symbolics to enable us to share the definitions between our stacks. System symbolics are being used in creating shared parmlib definitions for systems that are in a sysplex. With this facility, you use the symbols defined during system startup as variables in configuring your TCP/IP stack. This means that you only need to create and maintain a template file for all the systems in the sysplex.

System symbols processing

System symbols used in the PROFILE, OBEYFILE and INCLUDE files are automatically translated during stack initialization. However, if you are planning to use it on the other TCP/IP data sets such as TCPIP.DATA, you need to run a JCL-driven utility called EZACFSM1 located in hlq.SEZAINST(CONVSYM).

```

//CONVSYM JOB (accounting,information),programmer.name,
//          MSGLEVEL=(1,1),MSGCLASS=A,CLASS=A
//*
//STEP1 EXEC PGM=EZACFSM1,REGION=0K
//SYSIN DD DSN=TCP.DATA.INPUT,DISP=SHR
//*SYSIN DD PATH='/tmp/tcp.data.input'
//*          The input file can be either an MVS file or an HFS file.
//*
//*
//*
//SYSOUT DD DSN=TCP.DATA.OUTPUT,DISP=SHR
//*SYSOUT DD
// PATH='/tmp/tcp.data.output',PATHOPTS=(OWRONLY,OCREAT),
//*          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)

```

Figure 4-2 JCL for EZACFSM1

The input to EZACFSM1 is your template data set that contains the system symbols and the definitions that you need. The output data set will be the parameter files such as TCPIP.DATA, that the TCP/IP stack or CS for z/OS IP application will use during its startup and operation. You need to run the utility on each of the systems where you need to have the symbols translated.

Symbols definitions

The variable &SYSCONE is defined in the IEASYMxx member of SYS1.PARMLIB. As seen in Figure 4-3, the value for &SYSCONE is derived from &SYSNAME. The variable &SYSNAME could be defined either in the IEASYSxx member or in the LOADxx member used during IPL. In our case, &SYSNAME was defined in IEASYSxx, which we used to IPL our MVS images. Please look at Figure 4-4 for a sample of the IEASYSxx that we used for the startup of RA03. You can find further information about system symbols in *z/OS V1R1.0-V1R2.0 MVS Initialization and Tuning Guide, SA22-7591*.

```

SYSDEF          SYSCONE(&SYSNAME(3:2)) 1
                SYMDEF(&SYSR2='037RZ1')
                SYMDEF(&SYSR3='&SYSR2(1:5).2')
                SYMDEF(&SYSR4='&SYSR2(1:5).3')

```

Figure 4-3 &SYSCONE definition in SYS1.PARMLIB

```

PROG=(00,37,MQ),          AUTHORIZATION LIST
PROD=V2,
OMVS=04,                  OPEN EDITION
PLEXCFG=MULTISYSTEM,     SYSPLEX
COUPLE=XX,                SYSPLEX

PAGTOTL=(10,5),
REAL=0,
SMF=03,                   SELECT SMFPRM00, SMF PARAMETERS   DEFAULT
SSN=03,                   SUBSYSTEM INITIALIZATION NAMES
SYSNAME=RA03,             2   SAME AS SID IN SMFPRM00
VAL=00,                   SELECT VATLST00 DEFAULT

```

Figure 4-4 &SYSNAME definition in IEASYSxx

1 The value of SYSCclone is defined as two characters starting from the third character of SYSNAME.

2 SYSNAME is defined as RA03 for system RA03.

You can also define and use your own variable in configuring CS for z/OS IP aside from &SYSNAME or &SYSCclone.

System symbols in TCPIP.PROFILE

In Figure 4-5 we show the common configuration profile used for the A-stacks in systems RA03, RA28 and RA39. Since &SYSCclone is unique in each system, it ensures that the files and IDs that will be generated when the stacks initialize are also unique.

```

; MEMBER TCP.TCPPARMS(PROFILEA)
; *****
; THIS IS THE COMMON TCP/IP PROFILE FOR THE TCP/IP A-STACK IN
; SYSTEMS RA03, RA28 and RA39.
; *****
;
;
; DATASETPREFIX TCP
;
; TCPCONFIG
;
; UDPCONFIG
;
; *****
;
; IPCONFig
...
DYNAMICXCF 192.168.233.&SYSCLONE 255.255.255.0 1 1
;
AUTOLOG 1
;
; Start OMPROUTE in each of the A-Stacks
;
;   T&SYSCLONE.AOMPR      ; OSPF  SERVER  2
;
...
ENDAUTOLOG
;
; -----
PORT
....
135 UDP LLBD           ; NCS Location Broker
161 UDP T&SYSCLONE.SNMPD ; SNMP AGENT 3
162 UDP T&SYSCLONE.SNMPQ ; SNMP QUERY ENGINE
443 TCP OMVS          ; Domino webserver
...
;
; *****
; Include the stack and system specific device and home definitions. Devices
; are also started in the file.
;
; INCLUDE TCP.TCPPARMS(PRDV&SYSCLONE.A) 4
;
; *****
; Include Stack specific BSDRouting parameters for RouteD
;
; INCLUDE TCP.TCPPARMS(PR&SYSCLONE.ABSD)
;
...
;
; *****
;
; TCP.TCPPARMS(TELNETA1) contains the common definitions for all stacks.
; The VTAM parameters are in TCP.TCPPARMS(TELNETA1)
;
; INCLUDE TCP.TCPPARMS(TELNETA1) 5

```

Figure 4-5 Common TCP/IP configuration profile with system symbolics

1 Sets a unique IP address for the dynamic XCF definitions

2 Autologs the OMPR routing program

Note: A dot is needed at the end of &SYSCLONE because the next character is not a space.

3 Port reservation for each server

4 Include file for system specific device definitions

5 Common include file for TELNET and VTAM definitions

Important: The system symbols are stored in uppercase by MVS. Because you can code the TCP/IP configuration statements in either upper or lowercase, you have to make sure that you code the system symbol name in uppercase. If the symbols defined in the profile are not in uppercase, the symbols will not get translated and depending on the parameter, you may or may not get an error message but you would certainly encounter some unusual and unexpected names!

Include files

Together with the system symbolics support, we also used a facility introduced in OS/390 V2R5 IP to help us organize and share our stack configuration. By using the include configuration statement, we were able to structure our configuration better by putting different sections of PROFILE.TCPIP in separate files. During the stack's initialization, the contents of the file pointed to by the include statement are read and processed. These include statements are treated as if they were coded in PROFILE.TCPIP.

Because the devices used by each TCP/IP stack is unique, we used the variable &SYSCLONE (please refer to 4 in Figure 4-5) to resolve the name of the system-specific device file. We have included all of the DEVICE, LINK and START statements for each of the devices in this file. Figure 4-6 shows the device file for system RA03 called PRDV03A. We also have a separate device file each for systems RA28 and RA39.

```

; MEMBER TCP.TCPPARMS (PRDV03A)
; *****
; This file contains the stack specific device config data for system
; RA03 TCPIP A-Stack.
; *****
; VIPA Definition (For V2R7)
; *****
DEVICE VIPA3A  VIRTUAL  0
LINK  VIPA3A  VIRTUAL  0  VIPA3A
;
;
; *****
; LCS Definition osa ch 78          Device # 2060-2061
; *****
DEVICE TR1  LCS          2060 autorestart
LINK  TR1  IBMTR        0    TR1
...
;
HOME
    192.168.250.3  VIPA3A    ; 1st VIPA Link
    9.24.104.133  TR1
...
; *****
; Start all the defined devices.
;
START TR1
...
START FDDI1

```

Figure 4-6 Included device file PRDV03A for RA03

In our configuration, the three TCP/IP A-stacks were designed to have common Telnet and VTAM characteristics. This decision helped us to use and share the same file (5) in Figure 4-5) to define the Telnet and VTAM configuration for each of the TCP/IP stack. The variable &SYSCLONE is then used in the VTAM common configuration file to further customize the setting for each stack in the different systems. Figure 4-7 shows this common configuration file.

```

;*****
; MEMBER TCP.TCPPARMS(TELNETA1)
; Common TELNET and VTAM definition for the A-Stacks in systems
; RA03, RA28 and RA39
;*****

TELNETPARMS
;TESTMODE
    PORT 23
...
ENDTELNETPARMS

BEGINVTAM
PORT 23 223

...
; Define the LUs to be used for general users.
; NONE

    LUGROUP LU1
        RA&sysclone.TN01..RA&sysclone.TN05 1
    ENDLUGROUP
    PRTGROUP PRT1
        RA&SYSCLONE.TP01..RA&SYSCLONE.TP05
    ENDPRTGROUP
    LUMAP LU1 IP1 GENERIC PRT1
...

ENDVTAM

```

Figure 4-7 Common Telnet include file

Please take note of 1 in Figure 4-7. For illustration purposes, we used a different case for the &SYSCLONE variable and we got the error 2 in Figure 4-8 during the startup of the stack in RA03. Changing the case corrected the problem.

```

IEF403I  TCPIPA - STARTED - TIME=09.15.05
IEE252I  MEMBER CTIEZB01 FOUND IN SYS1.PARMLIB
EZZ7450I  FFST SUBSYSTEM IS NOT INSTALLED
EZZ0300I  OPENED PROFILE FILE DD:PROFILE
EZZ0309I  PROFILE PROCESSING BEGINNING FOR DD:PROFILE
EZZ0323I  AUTOLOG STATEMENT ON LINE 72 HAD NO ENTRIES
EZZ0300I  OPENED TCP.TCPPARMS(PRDV03A) FILE
EZZ0309I  PROFILE PROCESSING BEGINNING FOR TCP.TCPPARMS(PRDV03A)
EZZ0316I  PROFILE PROCESSING COMPLETE FOR FILE 'TCP.TCPPARMS(PRDV03A)'
EZZ0304I  RESUMING PROCESSING OF FILE DD:PROFILE
EZZ0300I  OPENED TCP.TCPPARMS(PRO3ABSD) FILE
EZZ0309I  PROFILE PROCESSING BEGINNING FOR TCP.TCPPARMS(PRO3ABSD)
EZZ0316I  PROFILE PROCESSING COMPLETE FOR FILE 'TCP.TCPPARMS(PRO3ABSD)'
EZZ0304I  RESUMING PROCESSING OF FILE DD:PROFILE
EZZ0300I  OPENED TCP.TCPPARMS(TELNETA2) FILE
EZZ0309I  PROFILE PROCESSING BEGINNING FOR TCP.TCPPARMS(TELNETA2)
EZZ0401I  LIST IS EMPTY IN FILE: 'TCP.TCPPARMS(TELNETA2)' ON LINE: 2
          57 AT: 'RA&SYSCLONE'
EZZ0401I  SYNTAX ERROR IN FILE: 'TCP.TCPPARMS(TELNETA2)' ON LINE: 2
          57 AT: 'RA&SYSCLONE'
EZZ0316I  PROFILE PROCESSING COMPLETE FOR FILE 'TCP.TCPPARMS(TELNETA2)

```

Figure 4-8 Startup error messages due to wrong case for &SYSCLONE

4.2.5 PROFILE.TCPIP parameters

The syntax for the parameters in the PROFILE TCPIP can be found in *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

Reserving ports

The PORT reservations that are defined in the PROFILE data set are the ports that are used by specific applications. You may decide to explicitly not reserve all well-known ports by defining the UNRESTRICTLOWPORTS option on the TCPCONFIG and UDPCONFIG statements. This would allow any socket application to acquire a well-known port.

```

TCPCONFIG
UNRESTRICTLOWPORTS

UDPCONFIG
UNRESTRICTLOWPORTS

```

Figure 4-9 PROFILE.TCPIP UNRESTRICTLOWPORTS statement

If you want the well-known ports to be used only by predefined application processes or superuser authorized application processes, then you can define the RESTRICTLOWPORTS option on the TCPCONFIG and UDPCONFIG statements. This prevents any non-authorized socket application from acquiring a well-known port. You then need to explicitly define PORT statements to reserve each port, or define the process with superuser authority in RACF. You may reserve the PORT or PORTRANGE by using the keyword OMVS, jobname of the process, or a wild card jobname such as *. UNIX applications may fork() another address space with a different name (for example, inetd or FTPD). You would need to reserve the port using the new address space name.

```

TCPCONFIG
  RESTRICTLOWPORTS

UDPCONFIG
  RESTRICTLOWPORTS

PORT
; 20 TCP T03FTP1 NOAUTOLOG ; FTP Server 2
  20 TCP OMVS NOAUTOLOG ; FTP Server 3
  21 TCP T03FTP1 4 ; FTP Server
; 23 TCP INTCLIEN ; Telnet Server
  25 TCP SMTP ; SMTP Server
; 23 TCP INTCLIEN ; Telnet Server
  25 TCP SMTP ; SMTP Server
  53 TCP T03DNS 1 ; Domain Name Server - Parent Process
  53 UDP T03DNS 1 ; Domain Name Server - Parent Process
  514 UDP OMVS 5 ; OE SyslogD Server
;
PORTRANGE 10000 2000 TCP OMVS ; TCP 10000 - 11999 7
PORTRANGE 10000 2000 UDP OMVS ; UDP 10000 - 11999 7
;

```

Figure 4-10 PROFILE.TCPIP: PORT & PORTRANGE

Normally you can specify either OMVS or the jobname in the PORT statement. However, certain daemons have special considerations on this matter.

When FTP starts it forks the listener process to run in the background, requiring that the name of the forked address space (T03FTP1 in this example), not the original procedure name, be used on the PORT statement of the control connection 4. You must specify OMVS as the name on the PORT for FTP's PORT 20 3, which is used for the data connection managed by the child process. If you specify the forked name on the data connection (Port 20, 2), the data connections will fail.

As you see, this can be a confusing issue. We recommend that you research each daemon that you implement; if something does not work, then try the alternative coding. You may read more about this subject in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840.

Please note that we reserve UDP port 514 5 to OMVS too. This port is used by the SyslogD server in OMVS to receive log messages from other SyslogD servers in the TCP/IP network.

In addition to assigning port numbers to servers, you also need to reserve the same range of ephemeral port numbers that was reserved on the NETWORK statement in BPXPRMxx.

7 These PORTRANGE statements reserve a range of ephemeral TCP and UDP ports for UNIX System Services. In this example, ports 10000 to 11999 are reserved. The range must match the INADDRANYPORT and INADDRANYCOUNT in your BPXPRMxx member 8.

```

NETWORK DOMAINNAME(AF_INET)
  DOMAINNUMBER(2)
  MAXSOCKETS(10000)

```

```

TYPE(CINET)
INADDRANYPORT(10000) 8
INADDRANYCOUNT(2000) 8

```

To display the PORT reservation list you can use the TSO/E command **NETSTAT PORTL**, MVS command **D TCPIP,procname,NETSTAT PORTL**, or UNIX shell command **onetstat -p t03atcp -o**.

Port#	Prot	User	Flags	Range
00007	TCP	MISCSERV	A	
00009	TCP	MISCSERV	A	
00019	TCP	MISCSERV	A	
00020	TCP	OMVS		
00021	TCP	T03AFTP1	A	
00025	TCP	SMTP	A	
00053	TCP	T03DNS	A	
00080	TCP	WEBQM	A	
00111	TCP	OMVS	A	
00520	UDP	T03AROU	A	
00580	UDP	NCPROUT	A	
00750	UDP	MVSKERB	A	
00751	UDP	ADM@SRV	A	
00760	UDP	IOASNMP	A	

Figure 4-11 Viewing port reservation list

Port sharing (TCP only)

If you want to run multiple instances of a listener for performance reasons, you can share the same port between them. TCP/IP will select the listener with the fewest connections (both active and in the backlog) at the time when a client request comes in. A typical application using this feature is the Internet Connection Secure Server. If the load gets high, additional servers are started by the Workload Manager.

Example of a shared port:

```

PORT
  80  TCP  WEBSRV1 SHAREPORT
  80  TCP  WEBSRV2
  80  TCP  WEBSRV3

```

BIND control for INADDR_ANY

A new keyword in the PORT statement was introduced in CS for OS/390 V2R10 IP: the BIND keyword. (CS for OS/390 V2R8 IP provides this support via APAR.) It associates the server jobname with a specific IP address when the server binds to INADDR_ANY. This new function can be used to change the BIND for INADDR_ANY to a BIND for a specific IP address.

Telnet, for example, is a server that binds to INADDR_ANY. Previously, an OS/390 client that wants to access both Telnet servers, Telnet 3270 and UNIX Telnet, would connect to different ports or different TCP/IP stacks, depending on which Telnet server it wished to connect to. This led to cases where either one server used a different, non-standard port or multiple TCP/IP stacks had to be used. With this function you do not need to have two different ports or TCP/IP stacks. You use the same port 23 for both Telnet 3270 and UNIX Telnet. All that is needed is to code the BIND keyword in the PORT statement for each server:

```
PORT
  23 TCP INTCLIEN BIND 172.16.251.7
  23 TCP OMVS BIND 172.16.251.8
```

In this case, the INTCLIEN is the special jobname associated with the TN3270 server, since it actually runs in the TCP/IP address space. The OMVS jobname identifies any OE server, including the UNIX Telnet server.

Both IP addresses can be dynamic VIPA addresses, static VIPA addresses or real interface addresses. You also can code a wild card for the jobname. Note that this function will work only for servers that bind to INADDR_ANY and is not valid with PORTRANGE statement.

AUTOLOG considerations

The purpose of the AUTOLOG statement is to start all procedures specified. AUTOLOG also monitors procedures started under its auspices, and will restart a procedure that terminates for any reason unless NOAUTOLOG is specified on the PORT statement.

For UNIX servers some special rules apply. If the procedure name on the AUTOLOG statement is eight characters long, no jobname need be specified. If the procedure name on the AUTOLOG statement is less than eight characters long and the job spawns listener threads with different names, you may have to specify the JOBNAME parameter and ensure that the jobname matches that coded on the PORT statement. In the following example, jobname T03FTP1 on the PORT statement matches JOBNAME on the AUTOLOG statement:

```
PORT
  20 TCP OMVS
  21 TCP T03FTP1

AUTOLOG 1
  T03FTP JOBNAME T03FTP1 ; FTP Server
ENDAUTOLOG
```

HOME

Starting with OS/390 V2R5 IP, the TCP/IP stack uses the IP address 127.0.0.1 for the loopback interface. If you also need to represent the IP address of 14.0.0.0 for compatibility with earlier MVS TCP/IP versions, you must code an entry in the HOME statement. The link label specified is LOOPBACK and you may define multiple IP addresses with the LOOPBACK interface.

```
HOME
  14.0.0.0 LOOPBACK
```

Figure 4-12 PROFILE.TCPIP HOME statement for LOOPBACK

The **onetstat -h** command displays the home address assignments of the currently running CS for z/OS IP system. It is similar in display to the **NETSTAT HOME** command in TSO, and the MVS command **D TCPIP,procname,NETSTAT,HOME**. There is an additional field, called the Flg field, that indicates which interface is the primary interface. The primary interface is the address that is inserted as the source address in an IP header when communicating to a destination through an indirect route. The primary interface is the first entry in the HOME list in the PROFILE.TCPIP definitions unless the PRIMARYINTERFACE parameter is specified.

```

$ onetstat -h

MVS TCP/IP onetstat CS/390 V2R7      TCPIP Name: T03ATCP
Home address list:
Address          Link          Flg
-----
192.168.251.1   LNKVIPA1     P
9.24.105.126    EN1
9.24.104.231    TR1
192.168.252.1   T03CTCP
192.168.100.100 LTR2
192.168.236.1   RAS
127.0.0.1       LOOPBACK

```

Figure 4-13 onetstat home display

IPCONFIG ARPTO

IPCONFIG ARPTO and ARPAGE statements have the same function: they specify the time interval between creation or revalidation and deletion of an entry in the ARP table. The value of IPCONFIG ARPTO is specified in seconds, and the value of ARPAGE is specified in minutes.

UNIX shell command **onetstat -R** displays the current ARP cache entries. The capital R in the option is required for this display. A third parameter may be coded that would specify the IP address of the entry you wish to display, as the **NETSTAT ARP ip_addr** command does from TSO. If you wish to display the entire ARP cache, you can specify the third parameter with the reserved word **ALL** (again, all in capital letters). If you do not specify in capital letters, the reserved word is not recognized. Figure 4-14 shows an example.


```

onetstat -p t03atcp -R ALL

MVS TCP/IP onetstat CS/390 V2R7      TCPIP Name: T03ATCP
Querying ARP cache for address 9.24.104.172

Link: TR1                IBMTR: 406137462144
Route info: 0620 5801 5810

Querying ARP cache for address 9.24.104.205
Link: TR1                IBMTR: 08005A5530E7
Route info: 0620 5803 5830

Querying ARP cache for address 9.24.104.188
Link: TR1                IBMTR: 0004AC6210AC
Route info: 0620 5804 5840

Querying ARP cache for address 9.24.104.1
Link: TR1                IBMTR: 400052005011
Route info: 0000

```

Figure 4-14 onetstat ARP display

IPCONFIG DYNAMICXCF

Starting with OS/390 V2R7 IP, you now have the option of either defining the DEVICE, LINK, HOME and START statements for MPC XCF connections to another z/OS or letting TCP/IP dynamically define them for you. Dynamic XCF devices and links, when activated, appear to the stack as though they had been defined in the TCP/IP profile. They can be displayed using standard commands, and they can be stopped and started. For multiple stack environments, IUTSAMEH links are dynamically created for same-LPAR links. Please refer to *Communications Server for z/OS V1R2 Implementation Guide Volume 5 : Availability, Scalability and Performance*, SG24-6517 for further details.

IPCONFIG PATHMTUDISCOVERY

Coding IPCONFIG PATHMTUDISCOVERY prevents the fragmentation of datagrams. It tells TCP/IP to discover dynamically the Path Maximum Transfer Unit (PMTU), which is the smallest of the MTU sizes of each hop in the path between two hosts.

When a connection is established, TCP/IP uses the minimum MTU of the sending host as the starting segment size and sets the Don't Fragment (DF) bit in the IP header. Any router along the route that cannot process the MTU will return an ICMP message requesting fragmentation and will inform the sending host that the destination is unreachable. The sending host can then reduce the size of its assumed PMTU. You can find more information about PMTU discovery in RFC 1191, Path MTU Discovery.

Aside from enabling PMTU during stack initialization, you could also enable or disable PMTU discovery by using **VARY OBEYFILE**.

IPCONFIG MULTIPATH

With the multipath feature of CS for z/OS IP, packets can now be load balanced on routes that have been defined to be of equal cost. These routes could either be learned dynamically or defined statically in your routing program (OMPROUTE or OROUTED). Without multipath support, all connections use the first active route to the destination network or host even if there are other equal-cost routes available. With multipath enabled, TCP/IP will select a route to that destination network or host on a round-robin basis. TCP/IP can select a route on a per-packet basis, but this is not recommended.

IPCONFIG IQDIOROUTING

Communications Server for z/OS IP V1R2 introduces performance improvements when routing IP traffic between HiperSockets (also known as Internal Queued Direct I/O or iQDIO) and Queued Direct I/O (QDIO). This type of routing is called *HiperSockets Accelerator* because it allows you to concentrate external network traffic over a single OSA-Express QDIO connection and then accelerates the routing over a HiperSockets link bypassing the TCP/IP stack. To enable HiperSockets Accelerator, code the IPCONFIG IQDIOROUTING parameter.

For further information on HiperSockets, see *Communications Server for z/OS V1R2 Implementation Guide Volume 4 : Connectivity and Routing*, SG24-6516 and *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775.

Storage usage

CS for z/OS IP uses Communications Storage Manager (CSM) to manage storage pools. The recommendation is to increase storage allocations by a minimum of 20 MB for TCP/IP in the CSA definition in IEASYSxx and the FIXED and ECSA definitions in IVTPRMxx.

New in CS for z/OS V1R2 are the GLOBALCONFIG ECSALIMIT and GLOBALCONFIG POOLLIMIT parameters. ECSALIMIT allows you to specify the maximum amount of extended common service area (ECSA) that TCP/IP can use, and POOLLIMIT allows you to specify the maximum amount of authorized private storage that TCP/IP can use within the TCP/IP address space. You can also use MVS command **D TCPIP,tcpproc,STOR** to display TCP/IP storage usage.

SACONFIG (SNMP subagent)

SACONFIG statement provides subagent support for SNMP. Through the subagent support you can manage an ATM OSA network interface. Please reference the PROFILE.TCPIP section in which you define SACONFIG:

```
SACONFig
  COMMUNity public      ; Community string
  OSASF 760             ; OSASF port number
; AGENT 161             ; Agent port number
  ENABLed
  SETSENAbled
  ATMENAbled
```

For more information on SNMP and subagent support, please see *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

SMFCONFIG

Prior to z/OS V1R2.0, all TCP/IP SMF records are written using record type 118. The type 118 record does not contain the identity of the TCP/IP stack. If you run multiple TCP/IP stacks, it is not easy to determine which SMF records relate to which TCP/IP stack. With z/OS V1R2.0, you have the option to record SMF type 119 records. Type 119 records contain additional values that identify the TCP/IP stack. Each type 119 record consists of an SMF header, a TCP/IP identification section, and a data section. The following type 119 records are available:

- ▶ TCP connection initiation and termination
- ▶ UDP socket close
- ▶ TCP/IP, interface and server port statistics
- ▶ TCP/IP stack start/stop
- ▶ FTP server transfer completion
- ▶ FTP server logon failure
- ▶ FTP client transfer completion
- ▶ TN3270 server session initiation and termination
- ▶ Telnet client connection initiation and termination

The SMFCONFIG statement is used to turn on SMF logging. It defines the type 118 and type 119 records to be collected (the default format is type 118).

The SMFPARMS statement can also be used to turn on SMF logging. However, you are encouraged to migrate to SMFCONFIG, which has the following advantages over the SMFPARMS statement:

- ▶ Using SMFCONFIG means that SMF records are written using standard subtypes. With SMFPARMS, you have to specify the subtypes to be used.
- ▶ SMFCONFIG allows you to record both type 118 and type 119 records. With SMFPARMS, only type 118 records can be collected.
- ▶ SMFCONFIG enables you to record a wider variety of information.
- ▶ By using SMFCONFIG, you gain support for dynamic reconfiguration, for all environments under which CS for z/OS IP is executing (SRB mode, reentrant, XMEM mode, etc.), and you can avoid duplicate SMF exit processes

In the following example, type 118 FTP client records, and type 119 TN3270 client records are collected:

```
SMFCONFIG TYPE118 FTPCLIENT
          TYPE119 TN3270CLIENT
```

The above example can also be coded this way:

```
SMFCONFIG FTPCLIENT
          TYPE119 TN3270CLIENT
```

since type 118 records are collected by default.

SMFCONFIG is coded in the PROFILE.TCPIP, but it has related entries in both Telnet and in FTP. (See Chapter 8, “TN3270 Telnet server” on page 175 and the FTP chapter in *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228 for more information on the associated coding.)

The only SMF exit supported in CS for z/OS IP is the FTP server SMF exit, FTPSMFEX. This exit is only called for type 118 records. If you need to access type 119 FTP SMF records, use the standard SMF exit facilities, IEFU83, IEFU84, and IEFU85.

For further information on TCP/IP SMF records, see *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775. *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 contains the SMF record layouts and the standardized subtype numbers used by Communications Server for z/OS IP.

ASSORTEDPARMS

If you use the ASSORTEDPARMS statement, consider migrating parameters from the ASSORTEDPARMS statement to the GLOBALCONFIG, IPCONFIG, TCPCONFIG and UDPCONFIG statements. All parameters on the ASSORTEDPARMS statement can be defined using the GLOBALCONFIG, IPCONFIG, TCPCONFIG and UDPCONFIG statements. Support for the ASSORTEDPARMS statement will be removed in a future release of CS for z/OS IP.

ASSORTEDPARMS should not be used with the GLOBALCONFIG, IPCONFIG, TCPCONFIG and UDPCONFIG statements, because unintended settings may result.

TCPCONFIG FINWAIT2TIME

The TCPCONFIG FINWAIT2TIME parameter allows you to specify the number of seconds a TCP connection should remain in the FINWAIT2 state. When this time limit is reached, the system waits a further 75 seconds before dropping the connection. The default is 600 seconds, but you can specify a value as low as 60 seconds, which will reduce the time a connection remains in the FINWAIT2 status, and thereby free up resources for future connections.

TCPCONFIG TCPTIMESTAMP

The TCP timestamp option is exchanged during connection setup. This option is enabled (by default) via the TCPCONFIG TCPTIMESTAMP parameter. Enabling TCP timestamp allows TCP/IP to better estimate the Route Trip Response Time (RTT), which helps avoid unnecessary retransmissions and helps protect against wrapping of sequence numbers.

4.3 Configuring the system with MVS commands

CS for z/OS IP provides a way to change the running TCP/IP configuration dynamically: the **VARY OBEYFILE** command. This command replaces the **OBEYFILE** TSO command in TCP/IP for MVS V3R2. The **VARY** command is an MVS Console command. It allows you to add, delete, or completely redefine all devices dynamically as well as change TN3270 parameters, routing, and almost any TCP/IP parameter in the profile. These changes are in effect until the TCP/IP started task is started again or another **VARY OBEYFILE** command overrides them. Authorization is through the user's RACF profile containing the MVS.VARY.TCPIP.OBEYFILE definition. There is no OBEY statement in the CS for z/OS IP PROFILE.TCPIP which, in earlier MVS TCP/IP implementations, provided authorization.

For further details on the **VARY OBEYFILE** command, see *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781. For more information on RACF definitions, please see *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840.

4.3.1 Deleting a device and adding/changing a device

You could use the **OBEYFILE** command to reconfigure the devices being used by CS for z/OS IP. Reconfiguration could either be deletion of existing devices, addition of new devices, or redefinition of an existing device. The syntax of the statements for OBEYFILE processing is the same as that being used in PROFILE.TCPIP.

Device reconfiguration is a three-step process:

1. Stop the device with an MVS console command (**VARY STOP**) or with a **VARY OBEYFILE** that names a data set in which the STOP command is defined.
2. Activate an OBEYFILE that deletes the link(s) and the device(s).
3. Activate an OBEYFILE that adds the new or changed link(s) and device(s) and then starts them.

If you wish to delete a device, the order of steps you take is important. The DELETE statement in PROFILE.TCPIP allows you to remove LINK, DEVICE and PORT or PORTRANGE definitions. The general sequence for deleting and adding back a device is:

1. Stop the device.
2. Remove the HOME address by excluding it from the full stack's HOME list.
3. Remove the Gateway statement pertaining to the link if you see error messages when you try to delete the link; do this by excluding it from the full gateway list of the stack.

Note: This step was required in OS/390 TCP/IP OpenEdition. It is not required in Communications Server for z/OS IP.

4. Delete the link.
5. Delete the device.
6. Add the new or changed device.
7. Add the new or changed link.
8. Add the HOME statements for the full stack.
9. Add the full gateway statements for the stack if you are using static routing.
10. Start the device.

Because the **STOP** command is executed as the last statement within an OBEYFILE regardless of its position within the file, you cannot do the STOP and DELETE in one step. Trying to do so will result in the error messages illustrated in Figure 4-15.

```
V TCPIP,T03ATCP,0,TCP.TCPPARMS(DELO3A)
EZZ0060I PROCESSING COMMAND: VARY TCPIP,T03ATCP,0,TCP.TCPPARMS(DELO3
A)
EZZ0300I OPENED OBEYFILE FILE 'TCP.TCPPARMS(DELO3A)'
EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCP.TCPPARMS(DELO3A)'
EZZ0395I DELETE LINK EN1 ON LINE 20 FAILED BECAUSE LINK IS ACTIVE
EZZ0395I DELETE DEVICE DEVEN1 ON LINE 21 FAILED BECAUSE DEVICE IS AC
TIVE
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCP.TCPPARMS(DELO3A)'
EZZ0303I OBEYFILE FILE CONTAINS ERRORS
EZZ0331I NO HOME ADDRESS ASSIGNED TO LINK EN1
EZZ0059I VARY OBEY COMMAND FAILED: SEE PREVIOUS MESSAGES
BPXF206I ROUTING INFORMATION FOR TRANSPORT DRIVER T03ATCP HAS BEEN
INITIALIZED OR UPDATED.
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE DEVEN1
```

Figure 4-15 Timing problems with device/link deletion

4.3.2 Example: changing an LCS device

In this example we wanted to change the Ethernet device at address 0306 from TYPE 802.3 to TYPE ETHERNET. This process would involve deleting the current definition and redefining the device. The results of our efforts follow.

In Figure 4-16, you see that there are 10 devices in the T03ATCP stack **1**. Under each device a single link is defined, which is associated with an IP address.

```
D TCPIP,T03ATCP,N,HOME
EZZ2500I NETSTAT CS V2R7 T03ATCP
HOME ADDRESS LIST:
ADDRESS      LINK          FLG
192.168.251.1 LNKVIPA1      P
9.24.105.126 EN1
192.168.252.1 T03CTCP
192.168.239.3 LINKT25A
192.168.202.18 LINK3746
192.168.221.20 LICP03
192.168.109.3 LICCP25
192.168.236.1 RAS
192.168.235.3 MPCT025
127.0.0.1    LOOPBACK
10 OF 10 RECORDS DISPLAYED 1

D TCPIP,T03ATCP,N,DEV
EZZ2500I NETSTAT CS V2R5 T03ATCP
DEVNAME: LOOPBACK      DEVTYPE: LOOPBACK  DEVNUM: 0000
LNKNAME: LOOPBACK      LNKTYPE: LOOPBACK  STATUS: READY
NETNUM: 0  QUESIZE: 0  BYTEIN: 0000012120  BYTEOUT: 0000012120
BSD ROUTING PARAMETERS:
MTU SIZE: 00000          METRIC: 00
DESTADDR: 0.0.0.0       SUBNETMASK: 0.0.0.0
DEVNAME: DEVVIPA1      DEVTYPE: VIPA      DEVNUM: 0000

.....

DEVNAME: DEVEN1        DEVTYPE: LCS        DEVNUM: 0306
LNKNAME: EN1          LNKTYPE: 8023 2  STATUS: READY
NETNUM: 0  QUESIZE: 0  BYTEIN: 0003131166  BYTEOUT: 0005792070
BROADCASTCAPABILITY: YES
BSD ROUTING PARAMETERS:
MTU SIZE: 01500          METRIC: 00
DESTADDR: 0.0.0.0       SUBNETMASK: 255.255.255.0

10 OF 10 RECORDS DISPLAYED 1
```

Figure 4-16 Displays of devices and home before deletion

Notice the link type of **2**: 8023. We need to change this in the running system by stopping, deleting, redefining, and adding back the Ethernet device and link.

In Figures 4-17, 4-18, and 4-20, you see the console messages that are issued as a result of these operations.

```

V TCPIP,T03ATCP,STOP,DEVEN1
EZZ0060I PROCESSING COMMAND: VARY TCPIP,T03ATCP,STOP,DEVEN1
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE DEVEN1
BPXF206I ROUTING INFORMATION FOR TRANSPORT DRIVER T03ATCP HAS BEEN
INITIALIZED OR UPDATED.

```

Figure 4-17 V TCPIP command to stop a device

Now that the device is stopped, we can issue the command to delete the device and its link:

```

V TCPIP,T03ATCP,0,TCP.TCPPARMS(DELO3A)
EZZ0060I PROCESSING COMMAND: VARY TCPIP,T03ATCP,0,TCP.TCPPARMS(DELO3A)
EZZ0300I OPENED OBEYFILE FILE 'TCP.TCPPARMS(DELO3A)'
EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCP.TCPPARMS(DELO3A)'
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCP.TCPPARMS(DELO3A)'
EZZ0053I COMMAND VARY OBEY COMPLETED SUCCESSFULLY
BPXF206I ROUTING INFORMATION FOR TRANSPORT DRIVER T03ATCP HAS BEEN
INITIALIZED OR UPDATED.

D TCPIP,T03ATCP,N,HOME
EZZ2500I NETSTAT CS V2R5 T03ATCP
HOME ADDRESS LIST:
ADDRESS          LINK          FLG
192.168.251.1    LNKVIPA1      P
192.168.252.1    T03CTCP
192.168.239.3    LINKT25A
192.168.202.18   LINK3746
192.168.221.20   LICP03
192.168.109.3    LICCP25
192.168.236.1    RAS
192.168.235.3    MPCT025
127.0.0.1        LOOPBACK
9 OF 9 RECORDS DISPLAYED

```

Figure 4-18 V TCPIP command to delete a device

The results of the delete are shown in the MVS NETSTAT console display of the HOME addresses. Notice the missing EN1 link as compared with Figure 4-16. In Figure 4-19, you see the statements necessary to delete the IP address, the LINK, and the DEVICE.

```

; From TCP.TCPPARMS(DELO3A)
;
; To remove Ethernet Link and Device (EN1)
HOME
  192.168.251.1   LNKVIPA1   ; 1st VIPA Link (for V2R5)
;;;9.24.105.126  EN1          ; 9.24.105.0
  192.168.252.1   T03CTCP    ; For SAMEHOST - IUTSAMEH - Connection
  192.168.239.3   LINKT25A
  192.168.202.18  LINK3746
  192.168.221.20  licp03
  192.168.109.3   liccp25
  192.168.236.1   RAS          ; XCF TO RA28
  192.168.235.3   MPCT025     ; MPC TO MVS25
DELETE LINK EN1
DELETE DEVICE DEVEN1
;

```

Figure 4-19 OBEYFILE member to delete a device (DEL03A)

We next add the device and link back with the changed definition **3**:

```

V TCPIP,T03ATCP,0,TCP.TCPPARMS(ADD03A)
EZZ0060I PROCESSING COMMAND: VARY TCPIP,T03ATCP,0,TCP.TCPPARMS(ADD03A)
EZZ0300I OPENED OBEYFILE FILE 'TCP.TCPPARMS(ADD03A)'
EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCP.TCPPARMS(ADD03A)'
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCP.TCPPARMS(ADD03A)'
EZZ0053I COMMAND VARY OBEY COMPLETED SUCCESSFULLY
BPXF206I ROUTING INFORMATION FOR TRANSPORT DRIVER T03ATCP HAS BEEN
INITIALIZED OR UPDATED.
EZZ4314I INITIALIZATION COMPLETE FOR DEVICE DEVEN1, LINK EN1

D TCPIP,T03ATCP,N,DEV
EZZ2500I NETSTAT CS V2R5

.....

DEVNAME: DEVEN1          DEVTYPE: LCS          DEVNUM: 0306
LNKNAME: EN1            LNKTYPE: ETH 3      STATUS: READY
NETNUM: 1  QUESIZE: 0   BYTEIN: 0000012092  BYTEOUT: 00000036
BROADCASTCAPABILITY: YES
BSD ROUTING PARAMETERS:
  MTU SIZE: 00000          METRIC: 00
  DESTADDR: 0.0.0.0       SUBNETMASK: 255.255.255.0
10 OF 10 RECORDS DISPLAYED

```

Figure 4-20 V TCPIP command to add a device

The member with the definition change in it is shown in Figure 4-21.


```

; From TCP.TCPPARMS(ADD03A)
;
; To add a new device/link pair
; ***** Top of Data *****
; *           3172-3 2nd floor           *
; *                                           *
; *****
;
;
DEVICE DEVEN1 LCS           306 NETMAN
LINK EN1 ETHERNET 1 DEVEN1
;
; -----
;
; HOME Internet (IP) addresses of each link in the host.
;
; NOTE: To use this home statement, update the ipaddress and linknames
;to reflect your installation configuration and remove the semicolon
;

; HOME Internet (IP) addresses of each link in the host.
HOME
192.168.251.1 LNKVIPA1 ; 1st VIPA Link (for V2R5)
9.24.105.126 EN1 ; 9.24.105.0
192.168.252.1 T03CTCP ; For SAMEHOST - IUTSAMEH - Connection
192.168.239.3 LINKT25A
192.168.202.18 LINK3746
192.168.221.20 licp03
192.168.109.3 liccp25
192.168.236.1 RAS ; XCF TO RA28
192.168.235.3 MPCT025 ; MPC TO MVS25
;
; -----

```

Figure 4-21 OBEYFILE member to add a device (ADD03A) (Part 1)

```

GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; Network First Hop Link Name Packet Size Subnet Mask Subnet Value

9          =          EN1          1500      0.255.255.0  9.24.105
192.168.252.2 =        T03CTCP      4096      HOST
192.168.202.8 =        LINK3746  DEFAULTSIZE  HOST
192.168.221   =        licp03       4000        0
192.168.239.27 =       LINKT25A      2000        host
192.168.109.2 =       liccp25       4000        HOST
192.168.236.2 =        RAS          32768      HOST
192.168.235.1 =       MPCT025      32768      HOST
;
;
; Indirect Routes - Routes that are reachable through routers on my
; network.
; Network First Hop Link Name Packet Size Subnet Mask Subnet Value
;
; Default Route - All packets to an unknown destination are routed
; through this route.
;
; Network First Hop Link Name Packet Size Subnet Mask Subnet Value

DEFAULTNET 9.24.105.1 EN1          1500      0
; *****
; Start all the defined devices.
START DEVEN1          ; 3172-3 ICP Ethernet

```

Figure 4-22 OBEYFILE member to add a device (ADD03A) (Part 2)

4.4 TCPIP.DATA

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

Notes:

- ▶ The syntax for the parameters in the TCPIP DATA file can be found in *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.
- ▶ A sample TCPIP.DATA config file is provided in *hlq.SEZAINST(TCPDATA)*.

The TCPIP.DATA file is also known as one of the *Resolver configuration* files. In fact, the name is now more commonly used to refer to this important file in the UNIX System Services environment because the sockets library contains a component called the Resolver. In the UNIX environment you use the */etc/resolv.conf* file for the same purpose as you use TCPIP.DATA in an MVS environment.

4.4.1 Resolvers

Resolver code executes as part of a socket program address space. It accesses name servers, on behalf of the application program, for name-to-address and address-to-name resolution. If no name server is available, the Resolver uses local definitions, such as `/etc/hosts`, `HOSTS.SITEINFO` or `HOSTS.ADDRINFO`. Statements in `TCPIP.DATA` tell the Resolver which name server, if any, it should use, and how to access that name server.

Prior to z/OS V1R2.0 there are several versions of the Resolver. The different versions of the Resolver use different search orders when trying to locate the `TCPIP.DATA` file; the search order for MVS socket applications is different to the search order for UNIX socket applications. It is therefore possible for applications using different socket APIs to use different `TCPIP.DATA` files.

With z/OS V1R2.0 the various Resolvers have been consolidated into a single Resolver, the System Resolver, which allows consistent name resolution processing across all applications.

System Resolver

The System Resolver is enabled automatically within z/OS. A Resolver address space is started during UNIX System Services initialization. The name of the address space is defined in `BPXPRMxx` with the `RESOLVER_PROC` statement. The address space reads the Resolver setup file, which contains two statements:

- ▶ `GLOBALTCPIPDATA` - specifies the name of an HFS file or MVS data set containing `TCPIP.DATA` statements that are to be used MVS image-wide.
- ▶ `DEFAULTTCPIPDATA` - specifies the name of an HFS file or MVS data set that will be used instead of data set `TCPIP.TCPIP.DATA` as the final location when searching for `TCPIP.DATA`.

The HFS file or MVS data set pointed to by the `GLOBALTCPIPDATA` statement must contain all `TCPIP.DATA` statements relating to the Resolver:

- ▶ `DOMAINORIGIN/DOMAIN`
- ▶ `NSINTERADDR/NAMESERVER`
- ▶ `NSPORTADDR`
- ▶ `RESOLVEVIA`
- ▶ `RESOLVERTIMEROUT`
- ▶ `RESOLVERUDPRETRIES`
- ▶ `SEARCH`
- ▶ `SORTLIST`

Other `TCPIP.DATA` statements can be specified in the global `TCPIP.DATA` file, or in a separate `TCPIP.DATA` file. If a statement, for example `TCPIPJOBNAME`, is not found in the global `TCPIP.DATA` file, the System Resolver will use either the MVS or UNIX search order, based on the Socket API in use, to locate the next `TCPIP.DATA` file, and this file will be examined for the statement `TCPIPJOBNAME`. When one `TCPIP.DATA` file is located, in addition to the global `TCPIP.DATA` file, the Resolver will stop searching, even if neither of the files contain `TCPIPJOBNAME` (if neither of the files contain `TCPIPJOBNAME`, the default value for `TCPIPJOBNAME` would be used). This effectively allows you to concatenate two files together to create the `TCPIP.DATA` configuration.

Resolver is also responsible for finding other files in the system. See Figure 4-23.

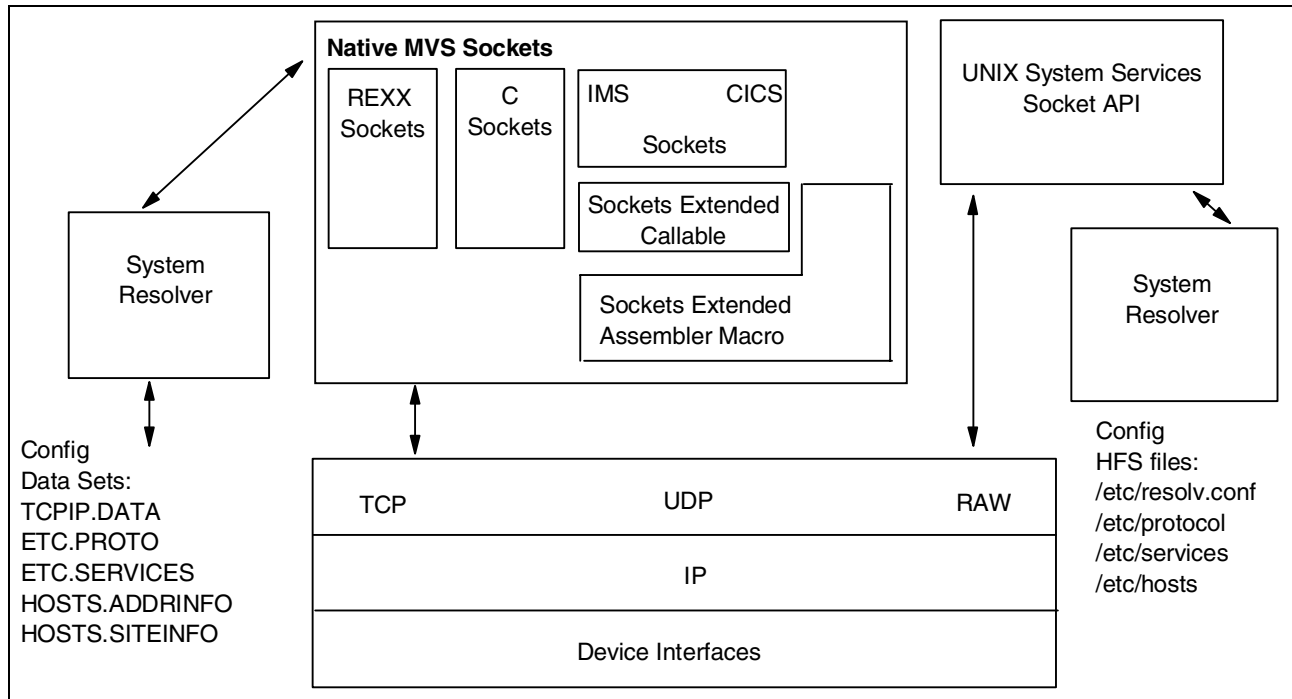


Figure 4-23 Resolver-related configuration files

Note: Though the System Resolver is a required feature, not making changes to your system will result in the same Resolver functionality as in previous releases: it uses the applicable MVS or UNIX search order, without the GLOBALTCPIP and DEFAULTTCPIP information.

For more information on the System Resolver, see *z/OS V1R2.0 CS: IP Migration*, GC31-8773, *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775, and *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

4.4.2 Resolver configuration for the TCP/IP stack

The functions invoked during stack initialization rely on the following search sequence for discovering the correct TCPIP.DATA information:

1. GLOBALTCPIPDATA
2. ENVAR("RESOLVER_CONFIG=...") (Explicit Allocation where the parm can be coded in either of the following fashions):
 - //TCPIP PROC PARMS='ENVAR("RESOLVER_CONFIG=/etc/tdata03a") ...
 - //TCPIP PROC
PARMS='ENVAR("RESOLVER_CONFIG=//"`TCP.TCPPARMS(TDATA03A)`")'
3. /etc/resolv.conf (HFS file) (Implicit Allocation)
4. //SYSTCPD DD DSN=TCP.TCPPARMS(TDAT03A) (Explicit Allocation)

Note: Some clients will have problems with the //SYSTCPD specification because the information from it is not passed to forked processes. For such clients, it is safer to use the RESOLVER_CONFIG variable or the HFS default of /etc/resolv.conf.

5. userid.TCPIP.DATA or jobname.TCPIP.DATA (Implicit Allocation)
6. SYS1.TCPPARMS(TCPDATA) (Implicit Allocation)
7. *h/q*.TCPIP.DATA (default = TCPIP.TCPIP.DATA) (Implicit Allocation) or DEFAULTTCPIPDATA

Note: Though not a result of the System Resolver, this search sequence represents a big change from earlier versions of TCP/IP. Note that //SYSTCPD is *fourth* in the search order, and no longer first.

The Telnet server, despite running in the TCP/IP address space, does not follow this search order. It uses the native MVS application search path.

4.4.3 MVS application search path

This search sequence is for most applications that are accessed through native MVS sockets. This is the path commonly used by programs written in the TCP/IP for MVS APIs, such as C, Sockets Extended, CICS, IMS, and REXX. TSO clients use this search sequence as well.

1. GLOBALTCPIPDATA
2. //SYSTCPD DD DSN=TCP.TCPPARMS(TDAT03A) (Explicit Allocation)
3. userid.TCPIP.DATA for TSO users or jobname.TCPIP.DATA for batch jobs (Implicit Allocation)
4. SYS1.TCPPARMS(TCPDATA) (Implicit Allocation)
5. TCPIP.TCPIP.DATA (Implicit Allocation) or DEFAULTTCPIPDATA

If one of the implicit methods for finding TCPIP.DATA can be used, TSO logon procedures need not be changed in order for TCP/IP clients to find the necessary configuration information. However, if you wish to make available the TCP/IP help information, you might consider adding the SEZAHHELP data set to the TSO logon procedure as depicted in **1** below.

```
* $RISC ACCOUNTING
//IKJACNT PROC
//GENERAL EXEC PGM=IKJEFT01,TIME=1440,REGION=4096K,DYNAMNBR=99,
//  PARM='EXEC ' 'ITSC.ISPF.CLISTS($RISC)''
.....
//SYSHELP DD DSN=SYS1.HELP,DISP=SHR
//          DD DSN=TCPIP.SEZAHHELP,DISP=SHR 1
.....
//*
```

Figure 4-24 TCP/IP help data in TSO logon procedure

4.4.4 z/OS UNIX application search path

z/OS UNIX applications are written using the UNIX System Services Socket API. Applications, such as those invoked from the UNIX System Services shell, would search for their TCPIP.DATA information in this sequence:

1. GLOBALTCPIPDATA
2. Any MVS data set or HFS file explicitly defined with a UNIX System Services environment variable called RESOLVER_CONFIG. This variable may be set by a UNIX System Services shell command, by passing it as a JCL PARM with the EXEC statement or with the STDENV DD card.
 - // PARM='ENVAR("RESOLVER_CONFIG=/etc/tdata03a")'...
 - // PARM='ENVAR("RESOLVER_CONFIG=//"**TCP.TCPPARMS(TDATA03A)**")' '
 - //STDENV DD

UNIX applications that use BPXBATCH to get started, as for example ORouted, can use the special //STDENV DD card to point to a file that contains the environmental variables. BPXBATCH will read this file and set the variables before starting the UNIX program.

 - RESOLVER_CONFIG=//"**TCP.TCPPARMS(TDATA03A)**" (ISHELL)
 - RESOLVER_CONFIG=/etc/tdata03a (ISHELL)
 - export RESOLVER_CONFIG=/etc/tdata03a (OMVS shell)
 - export RESOLVER_CONFIG=//"**TCP.TCPPARMS(TDATA03A)**" (OMVS shell)

See Figure 4-29 on page 104 for another example.
3. /etc/resolv.conf (HFS file) (Implicit Allocation)
4. //SYSTCPD DD DSN=**TCP.TCPPARMS(TDATA03A)** (Explicit Allocation)

This option may not be a good technique for processes that use the fork() command. This allocation will not be available to the child process that is forked since DD allocations for the parent process are not inherited by the child. The only exception to this rule is a STEPLIB allocation.
5. userid.TCPIP.DATA (Implicit Allocation)
6. SYS1.TCPPARMS(TCPDATA) (Implicit Allocation)
7. TCPIP.TCPIP.DATA (Implicit Allocation) or DEFAULTTCPIPDATA

4.4.5 Working with TCPDATA

Applications using CS for z/OS IP can override current TCPIP.DATA settings using the RESOLVER_CONFIG environment variable. This, however, will not override the name server specification if a GLOBALCONFIG file was specified.

From an OMVS shell environment

If an application wants to override the system default of /etc/resolv.conf, it can initialize the RESOLVER_CONFIG environment variable to point to an alternate Resolver configuration file or data set.

You might want to override the default if, for example, you have a requirement to use different name servers. If you are logged on to the z/OS UNIX shell, the **export** command can be used to point to an MVS data set:

```

-----
# export RESOLVER_CONFIG="//'TCP.TCPPARMS(TDATATST)'"
# echo $RESOLVER_CONFIG
//'TCP.TCPPARMS(TDATATST)'
#
#
Command ==>                               Scroll ==

```

Figure 4-25 Overriding TCPDATA from OMVS shell: MVS data set

Please note the required syntax for referring to an MVS data set name: double quotes followed by two slashes followed by the MVS data set name enclosed in single quotes. You will have to use the same syntax for other environment variables that point to MVS data sets.

The following shell command can be used to point to an HFS file:

```

-----
# export RESOLVER_CONFIG=/etc/resolv.conf.tst3a
# echo $RESOLVER_CONFIG
/etc/resolv.conf.tst3a
#
#
Command ==>                               Scroll ==

```

Figure 4-26 Overriding TCPDATA from OMVS shell: HFS file

From ISHELL

An idiosyncrasy of the ISHELL environment is that any dynamic changes to the environment, such as a temporary change of TCPDATA, must be made for every command that is entered. A change to a new RESOLVER_CONFIG, as shown in Figure 4-27, is only temporary.

```

File Directory Special_file Tools File_systems Options Setup Help
-----+
Enter a Shell Command
-----+
Enter a shell command and press Enter.

Standard output and standard error are redirected to a temporary
file. If there is any data in the file when the shell command
completes, the file is displayed.
  RESOLVER_CONFIG="//'TCP.TCPPARMS(TDATATST)'" onslookup sys03
  _____
  _____
  _____

F1=Help      F3=Exit      F6=Keyshelp  F12=Cancel
-----+

```

Figure 4-27 Changing TCPDATA environment for ISHELL client

The corresponding format for an HFS Resolver configuration file under the ISHELL is:

```
RESOLVER_CONFIG=/etc/resolv.conf.tst3a onslookup sys03
```

From TSO

TCP/IP client programs that execute under TSO must also be able to locate an explicitly allocated TCPIP.DATA data set in order to use a non-default Communications Server for z/OS IP stack. You can allocate a TCPIP.DATA data set to a TSO user in two ways:

1. Add a SYSTCPD DD statement to your TSO logon procedure. The issue with this approach is that you will need a separate TSO logon procedure per CS for z/OS IP stack in your MVS system, and users will have to log off TSO and log on again with another TSO logon procedure in order to switch from using one CS for z/OS IP stack to another.
2. Use one common TSO logon procedure without a SYSTCPD DD statement. Before a TSO user starts any TCP/IP client programs, the user has to execute a TSO **ALLOC** command to allocate a TCPIP.DATA data set to DD-name SYSTCPD. To switch from using one CS for z/OS IP stack to another, the user simply has to deallocate the current SYSTCPD allocation and allocate another TCPIP.DATA data set. (See Chapter 5, "Multiple TCP/IP stacks on z/OS" on page 117 for more details on how to operate in a multiple TCP/IP stack environment.)

This method is very convenient, especially if you are trying to test different values in your TCPIP.DATA file. You may even include the switch to different TCPIP.DATA configuration in a REXX CLIST. R Figure 4-28 is an example of freeing and allocating a new TCPIP.DATA file.

```
free fi(SYSTCPD)
alloc fi(SYSTCPD) da('tcp.tcparms(tdatatst)') SHR
```

Figure 4-28 Changing TCPDATA environment for TSO client

From JCL startup procedure

To define the RESOLVER_CONFIG variable in a JCL startup procedure, you would pass the information as a PARM using the ENVAR reserved word. A sample of the JCL coding for this definition is shown in Figure 4-29.

```
//T03FTPD PROC PARMS='ENVAR("RESOLVER_CONFIG=/etc/resolv.conf.tst3a")'
//T03FTP EXEC PGM=FTPD,REGION=OM,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON)/&PARMS'
```

Figure 4-29 Sample JCL for RESOLVER_CONFIG variable to an HFS file

An example of setting the RESOLVER_CONFIG variable to an MVS data set in a JCL procedure is shown in Figure 4-30.

```
//T03ATCP PROC PARMS='CTRACE(CTIEZB01)',
// XS='ENVAR("RESOLVER_CONFIG=//TCP.TCPPARMS(TDATA03A)")'
//TCPIP EXEC PGM=EZBTCPIP,
// PARM='&PARMS &XS',
// REGION=7M,TIME=1440
```

Figure 4-30 Sample JCL for RESOLVER_CONFIG variable to an MVS data set

From the STDENV file

If the application is started with BPXBATCH the STDENV DD statement can be used to point to a file containing the environment variables. A sample of the contents of this file is in Figure 4-31.


```

RESOLVER_CONFIG=/etc/resolv.conf.3a
ROUTED_PROFILE=/'TCP.TCPPARMS(RD03APR)'
GATEWAYS_FILE=/'TCP.TCPPARMS(RD03AGW)'
```

Figure 4-31 Sample STDENV file for ORouteD

4.4.6 Testing TCPIP.DATA

HOMETEST is a TSO command that can be used to test your TCPIP.DATA specifications. It is meant to be issued from TSO only, so it uses the native MVS search order and Resolver when locating configuration data sets and/or doing name to IP address resolutions. Figure 4-32 shows an example.

```

EZA0619I Running IBM MVS TCP/IP CS/390 V2R7 TCP/IP Configuration Tester

EZA0620I The TCP/IP system parameter file used will be SYSTCPD DD.

EZA9431I FTP.DATA file not found. Using hardcoded default values.

EZA0602I TCP Host Name is: mvs03.itso.ral.ibm.com

EZA0605I Using Name Server to Resolve mvs03.itso.ral.ibm.com
EZA9455I TCP Host Name:           MVS03A
EZA9456I Domain Origin:          itso.ral.ibm.com
EZA9457I Jobname of TCP/IP:      T03ATCP
EZA9458I Communicate Via:        UDP
EZA9462I OpenTimeOut:           30
EZA9463I MaxRetrys:              2
EZA9464I NSPort:                 53
EZA9465I NameServer Jobname:     NAMESRV
EZA9466I NSInternetAddress(.1.) := 192.168.250.3
EZA9482I Data Set Prefix used:   TCP
EZA9468I

EZA9469I Resolving Name:         MVS03A.itso.ral.ibm.com
EZA9470I Result from InitResolver: OK
EZA9471I Building Name Server Query:
EZA9554I * * * * * Beginning of Message * * * * *
EZA9555I Query Id:               1
EZA9556I Flags:                  0000 0001 0000 0000
EZA9516I Number of Question RRs: 1
EZA9517I Question 1: MVS03A.itso.ral.ibm.com A (9486) IN (9507)
EZA9516I Number of Answer RRs:   0
EZA9516I Number of Authority RRs: 0
EZA9516I Number of Additional RRs: 0
```

Figure 4-32 Testing TCPIP.DATA with HOMETEST (Part 1)

```

EZA9474I HostNumber (1) is: 192.168.250.3
EZA0611I The following IP addresses correspond to TCP Host Name:
          MVS03A.itso.ral.ibm.com
EZA0612I 192.168.250.3
EZA0614I The following IP addresses are the HOME IP addresses defined in
          PROFILE.TCPIP:
EZA0615I 192.168.233.3
EZA0615I 192.168.233.3
EZA0615I 192.168.233.3
EZA0615I 9.24.104.113
EZA0615I 192.168.250.3
EZA0615I 192.168.125.1
EZA0615I 192.168.125.3
EZA0615I 192.168.221.7
EZA0615I 9.24.105.76
EZA0615I 192.168.221.3
EZA0615I 192.168.229.3
EZA0615I 192.168.20.3
EZA0615I 192.168.100.100
EZA0615I 192.168.235.3
EZA0615I 192.166.236.1
EZA0615I 127.0.0.1

EZA0618I All IP addresses for MVS03A.itso.ral.ibm.com are in the HOME list!

EZA0622I Hometest was successful - all Tests Passed!
***

```

Figure 4-33 Testing TCPIP.DATA with HOMETEST (Part 2)

4.5 Configuring the SITE table (HOSTS.LOCAL)

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

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

If you have configured your Resolver to use a name server, it will always try to do so, unless your applications were written with a RESOLVE_VIA_LOOKUP symbol in the source code. If this is the case, all name resolution calls from such a program will always use the local hosts file. Additionally, the LOOKUP keyword in TCPIP.DATA will result in first attempting to look up a name in the local hosts file and then using the name server.

It may also be a good idea to have some basic local hosts file available for the Resolver to use if the name server is not reachable. If the name server does not respond to name resolution requests, the Resolver will try to use the local hosts file.

If the name server is reachable but returns a negative reply for a name resolution request, the Resolver will try to resolve the unqualified name via the local hosts file, if such a file is present. Assume you try to resolve the host name *friendly* and your DOMAINORIGIN is *my.wood.com*, the Resolver will send a query to the name server for *friendly.my.wood.com*. If the name server returns a negative reply (the name is not registered), the Resolver will look into the local hosts file for an entry of *friendly* and, if not found, for an entry of *friendly.my.wood.com*.

Due to the higher flexibility of the Domain Name System, we recommend you use a domain name server. If you have Communications Server for z/OS IP installed, you can configure and use the name server that comes with it. You need to configure your Resolver configuration file to point to the IP host in which your name server is running. This may be any host that is reachable from your UNIX System Services environment. There is no requirement that the name server must run on z/OS unless you wish to take advantage of DNS/WLM provided with Communications Server for z/OS IP. Even if you use DNS/WLM of CS for z/OS IP for a sysplex subdomain, you may continue to use another DNS for your parent domain. See more about this topic in *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228 and *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date July 2002)).

Note: DNS/WLM support is only available with the BIND4 DNS server; it is currently not supported in BIND9.

If you set up a small TCP/IP network, the simplicity of the local hosts file approach is preferable.

The Resolver will attempt to service the following calls first via a request sent to a name server, and then via the local hosts file:

Gethostbyname Resolve a host name into one or more IP addresses
Gethostbyaddr Resolve an IP address into a host name

The following Resolver calls only use the local hosts file. If you have configured your system to use a name server, these calls will bypass the name server and use your local hosts file, if you have configured one. If no local hosts file exists, the calls will return with an error.

Sethostent Prepare to read your local hosts file sequentially
Gethostent Read next entry in your local hosts file
Endhostent End reading your local hosts file sequentially

CS for z/OS IP supports two different formats for the local hosts file:

- ▶ The standard BSD formatted text file - as it is supported in most TCP/IP implementations, implemented via HFS file `/etc/hosts`.
- ▶ The format that the Communications Server for z/OS IP MAKESITE utility program creates. The MAKESITE utility program comes with Communications Server for z/OS IP.

You can use either format in a CS for z/OS IP environment, but note the following:

- ▶ Applications that use MVS sockets must use the format created via the MAKESITE utility
- ▶ The following Resolver calls need the format created via the MAKESITE utility:

Getnetbyaddr Get a net entry by name
Getnetbyname Get a net entry by network address

Setnetent	Prepare to read the net entries sequentially
Getnetent	Get next net entry
Endnetent	End reading net entries sequentially

In most UNIX systems these calls are serviced via a file called `/etc/networks`, but this file is currently not supported by CS for z/OS IP. If you use the Communications Server for z/OS IP **MAKESITE** utility, this utility supports a `HOSTS.LOCAL` source file according to the RFC952 syntax, which allows you to specify both host and network entries. The resulting files from **MAKESITE**, `HOSTS.ADDRINFO` and `HOSTS.SITEINFO` may therefore hold both host and network entries, which is the reason why these calls are supported if you use the **MAKESITE** format.

The z/OS UNIX search order for `HOSTS.SITEINFO` is as follows:

1. The value of the environment variable `X_SITE`.
This should point to the `HOSTS.SITEINFO` MVS data set that was created by the **MAKESITE** command. This should *not* refer to an HFS file, because the two types of data are incompatible.
2. `/etc/hosts` file that resides in the HFS.
3. `userid.HOSTS.SITEINFO` for TSO
4. `datasetprefix.HOSTS.SITEINFO`
where `datasetprefix` represents the value of the `DATASETPREFIX` keyword specified in `TCPIP.DATA` configuration file. The default is `TCPIP`.

The z/OS UNIX search order for `HOSTS.ADDRINFO` is as follows:

1. The value of the environment variable `X_ADDR`.
This should point to the `HOSTS.ADDRINFO` MVS data set that was created by the **MAKESITE** command.
2. `/etc/hosts` file that resides in the HFS. This step is skipped unless the request is a `gethostbyaddr()`.
3. `userid.HOSTS.ADDRINFO`
4. `datasetprefix.HOSTS.ADDRINFO`
`datasetprefix` represents the value of the `DATASETPREFIX` keyword specified in `TCPIP.DATA` configuration file. The default is `TCPIP`.

The native MVS Sockets search order for data sets `HOSTS.ADDRINFO` and `HOSTS.SITEINFO` is:

1. `jobname.HOSTS.xxxxINFO` for batch jobs or `userid.HOSTS.xxxxINFO` for TSO users
2. `datasetprefix.HOSTS.xxxxINFO`
where `datasetprefix` represents the value of the `DATASETPREFIX` keyword specified in the `TCPIP.DATA` configuration file. The default is `TCPIP`.

For host names and address information you can use either the `HOSTS.SITEINFO` and `HOSTS.ADDRINFO` data sets built by **MAKESITE**, or HFS file `/etc/hosts`. When you use `/etc/hosts` it can supply both name-to-address and address-to name resolution. `HOST.SITEINFO` supplies name-to-address resolution, and `HOST.ADDRINFO` supplies address-to-name resolution.

4.5.1 /etc/hosts

If you want to use the text format, we recommend that you place your local hosts file in the Hierarchical File System under the name `/etc/hosts` (the standard location). Please observe that some of the available documentation specifies this name differently (without the last "s"), which will not work.

See Figure 4-34 for a sample `/etc/hosts` file.

```
#
# OE Resolver /etc/hosts file
#
# The format of this file is:
#
# Internet Address      Host name  Aliases   # Comments
#
# Items are separated by any number of blanks and/or tabs. A '#'
# indicates the beginning of a comment; characters up to the end of the
# line are not interpreted by routines which search this file. Blank
# lines are allowed in this file.

9.24.104.126   mvs18oe mvs0e # OE host
192.168.210.1  mvs18an      # AnyNet MVS host
192.168.210.8  mypcaa       # AnyNet gw host
9.24.104.79    mypc         # A workstation
```

Figure 4-34 Sample `/etc/hosts` file

There are some syntax requirements for the `/etc/hosts` file. The most important are the following:

1. A host name can have a maximum of four qualifiers:
 - Host name `a.b.c.d` is a valid host name.
 - Host name `a.b.c.d.e` is *not* a valid host name.
2. You can specify a maximum of 35 aliases per IP address.
3. A qualifier can have a maximum length of 24 characters:
 - Host name `mypchost.mynet` is a valid host name.
 - Host name `myotherpchost.mynet` is a valid host name.
 - Host name `myotherpchostinlosangeles.mynet` is *not* a valid host name.

If you already have a CS for z/OS IP stack implemented and want to maintain your local hosts file in one place, you can instruct the UNIX Resolver to use the same file format as CS for z/OS IP uses.

You can use two different approaches:

1. Maintain your local hosts file in the source format that is accepted by the TCP/IP for MVS MAKESITE utility program. This format is documented in RFC952.
2. Maintain your local hosts file in the BSD source format that is accepted by the UNIX Resolver and most other TCP/IP platforms.

4.5.2 Maintaining shared source in HOSTS.LOCAL

You maintain your `HOSTS.LOCAL` file in the format that is required by the TCP/IP for MVS MAKESITE utility program. See Figure 4-35 for a sample `HOSTS.LOCAL` data set.

```

; HOSTS.LOCAL (Input to TCP/IP for MVS MAKESITE utility)
;
; Syntax requirements documented in RFC952.
;
HOST : 9.24.104.126 : mvs18a, mvsoe :::
HOST : 192.168.210.1 : mvs18aa :::
HOST : 192.168.210.8 : mypcaa :::
HOST : 9.24.104.79 : mypc :::
HOST : 9.24.104.80 : abc.ibm.com :::
HOST : 9.24.104.81 : abc1, abc1.ibm.com :::
;
NET : 9.24.104.0 : itso.ra1.ibm.com :
NET : 9.0.0.0 : ibm.com :

```

Figure 4-35 Sample HOSTS.LOCAL source

When you run the MAKESITE utility program, it produces two output data sets:

- ▶ datasetprefix.HOSTS.SITEINFO
- ▶ datasetprefix.HOSTS.ADDRINFO

You can instruct the UNIX Resolver to use these data sets in two ways:

1. To use these data sets as a system default, you must ensure that there is no file called /etc/hosts in your Hierarchical File System, and in addition, you must specify the DATASETPREFIX keyword in your TCPIP.DATA file.

If your DATASETPREFIX in the TCPIP.DATA configuration data set or file is TCPIP.OMVS, the UNIX Resolver will use:

```

TCPIP.OMVS.HOSTS.SITEINFO
TCPIP.OMVS.HOSTS.ADDRINFO

```

2. If you want to override your system default for a specific application, you can set the two environment variables called X_SITE and X_ADDR to point to two data sets that are created with the MAKESITE utility. If your application executes in the shell environment, you can use the following commands to assign values to the environment variables:

```

export X_SITE="//'MYOWN.HOSTS.SITEINFO'"
export X_ADDR="//'MYOWN.HOSTS.ADDRINFO'"

```

If you use the same DATASETPREFIX for your TCP/IP for MVS Resolver, the same set of HOSTS.SITEINFO and HOSTS.ADDRINFO data sets can be used by both MVS and z/OS UNIX socket applications.

4.5.3 Maintaining shared source in /etc/hosts

Though CS for z/OS IP can now directly read the /etc/hosts file, you may wish to create HOSTS.SITEINFO and HOSTS.ADDRINFO data sets from your /etc/hosts source file. Because the format of the /etc/hosts file is not compatible with the format that is required by the CS for z/OS IP MAKESITE utility, you would need to process the /etc/hosts file with a small home-written REXX program before running the MAKESITE command. The sample REXX program in Appendix A, “Sample REXX to create HOSTS.LOCAL from /etc/hosts” on page 221 can be used for such a purpose.

4.6 /etc/protocol

HFS file /etc/protocol and MVS data set ETC.PROTO hold information about supported protocols. The /etc/protocol file is used by the following Resolver calls:

Getprotobyname	Get protocol number based on a protocol name
Getprotobynumber	Get protocol name based on a protocol number
Setprotoent	Prepare to read the protocol file sequentially
Getprotoent	Read next protocol file entry
Endprotoent	End reading the protocol file sequentially

The protocol number is used in the 8-bit protocol field in an IP packet header.

To create HFS file /etc/protocol, we used the sample protocol member from datasetprefix.SEZAINST(PROTO), which contains the following (some comments removed for readability):

```
# official name, protocol number, aliases
ip          0          # dummy for IP
icmp       1          # control message protocol
tcp        6          # tcp
udp        17         # user datagram protocol
```

Figure 4-36 /etc/protocol sample

The z/OS UNIX search order for the protocol file is as follows:

1. /etc/protocol that resides in the HFS.
2. *userid*.ETC.PROTO
3. datasetprefix.ETC.PROTO

datasetprefix represents the value of the DATASETPREFIX keyword specified in TCPIP.DATA configuration file. The default is TCPIP.

The MVS Sockets search order for the protocol file is:

1. *jobname*.ETC.PROTO for batch jobs or *userid*.ETC.PROTO for TSO users
2. datasetprefix.ETC.PROTO

where datasetprefix is the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file. The default is TCPIP.

4.7 /etc/services

HFS file /etc/services and MVS data set ETC.SERVICES hold information about how individual applications are assigned to port numbers. Standard applications, such as telnet or FTP, are assigned port numbers inside the well-known port number range (from 0 to 1023). You can assign port numbers to your own server applications by adding entries to the /etc/services file. The content of /etc/services is typically used by a server program via a getservbyname() call, where the server program passes its own name and receives the assigned port number when the call returns. The server program can then bind its socket to

the assigned port number. This technique allows you to keep port number assignments external to your server program. This is of particular importance if you want to start more instances of your server program on the same TCP/IP stack. By using different /etc/services files, each instance of the server program may be assigned alternate port numbers.

The /etc/services file is used by the following Resolver calls:

- Getservbyname** Get server port number based on server name
- Getservbynumber** Get server name based on server port number
- Setservent** Prepare to read /etc/services sequentially
- Getservernt** Read next entry in /etc/services
- Endservent** End reading /etc/services sequentially

The port number is used on various socket calls and is also included in both the header of a TCP segment and the header of a UDP datagram.

Figure 4-37 shows an extract of the /etc/services file.

# Name	Port/protocol	Aliases	
echo	7/tcp		
echo	7/udp		
discard	9/tcp	sink null	
discard	9/udp	sink null	
systat	11/tcp	users	
daytime	13/tcp		
daytime	13/udp		
netstat	15/tcp		
qotd	17/tcp	quote	
chargen	19/tcp	ttytst source	
chargen	19/udp	ttytst source	
ftp	21/tcp		
telnet	23/tcp		
smtp	25/tcp	mail	
time	37/tcp	timserver	
time	37/udp	timserver	
rlp	39/udp	resource	# resource location
nameserver	42/tcp	name	# IEN 116
whois	43/tcp	nickname	
domain	53/tcp	nameserver	# name-domain server
domain	53/udp	nameserver	

Figure 4-37 /etc/services sample (extract)

In general, servers use a getservbyname() call to find the assigned port number. Some servers allow you to override the port number via a server-specific configuration or start option. The FTPD server, for example, allows you to pass a runtime option:

```
PORT 7021
```

If this option is specified, the FTPD server will not use the value assigned in /etc/services, but use the value specified in the runtime option. You can use this technique to start alternate FTPD server instances on alternate port numbers.

Servers that are started via InetD use the service name, which you specify in the `/etc/inetd.conf` file as argument on the `getservbyname()` call. If you, for example, want to have your UNIX TelnetD server to operate on port 2023 instead of the default port 23, you can edit the line in your `/etc/services` file that corresponds to the Telnet service name in your `/etc/inetd.conf` file.

```
/etc/inetd.conf (extract):
-----
telnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetsd otelnetd

/etc/services (extract):
-----
telnet  2023/tcp
```

Figure 4-38 Assigning port number to the TelnetD server

If you want to start two Telnet servers in your UNIX environment, you can use the following set of definitions:

```
/etc/inetd.conf (extract):
-----
telnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetsd otelnetd
testtelnet stream tcp nowait OMVSKERN /usr/test/otelnetsd otelnetd

/etc/services (extract):
-----
telnet  23/tcp
testtelnet 2023/tcp
```

Figure 4-39 Assigning port numbers to two TelnetD servers

With this setup, you will have your normal UNIX Telnet server operating on the standard port 23, and you will have your test Telnet server operating on port 2023. If a telnet client connects to port 2023, the telnet client will use your test Telnet server.

The z/OS UNIX search order for the services file is as follows:

1. `/etc/services` that resides in the HFS
2. `userid.ETC.SERVICES`
3. `datasetprefix.ETC.SERVICES`

`datasetprefix` represents the value of the `DATASETPREFIX` keyword specified in the `TCPIP.DATA` configuration file. The default is `TCPIP`.

The MVS search order for the services file is:

1. The `//SERVICES` DD name
2. `jobname.ETC.SERVICES` for batch jobs or `userid.ETC.SERVICES` for TSO users
3. `datasetprefix.ETC.SERVICES`

where datasetprefix represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file. The default is TCPIP.

4.8 Starting Communications Server for z/OS IP

Figure 4-40 shows the messages issued during the start of Communications Server for z/OS IP.

```
S T03ATCP
IEF695I START T03ATCP WITH JOBNAME T03ATCP IS ASSIGNED TO USER TCPIP3
, GROUP OMVSRP
$HASP373 T03ATCP STARTED
IEF403I T03ATCP - STARTED - TIME=08.38.06
IEE252I MEMBER CTIEZB01 FOUND IN SYS1.PARMLIB 1
EZZ0300I OPENED PROFILE FILE DD:PROFILE
EZZ0309I PROFILE PROCESSING BEGINNING FOR DD:PROFILE
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE DD:PROFILE 2
EZZ0334I IP FORWARDING IS ENABLED
EZZ0335I ICMP WILL IGNORE REDIRECTS
EZZ0350I SYSPLEX ROUTING SUPPORT IS ENABLED
EZZ0351I SOURCEVIPA SUPPORT IS DISABLED
EZZ0352I VARIABLE SUBNETTING SUPPORT IS DISABLED
EZZ0345I STOPONCLAWERROR IS ENABLED

BPXF206I ROUTING INFORMATION FOR TRANSPORT DRIVER T03ATCP HAS BEEN
INITIALIZED OR UPDATED. 3
EZZ4202I OPENEDITION-TCP/IP CONNECTION ESTABLISHED FOR T03ATCP 4
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE RAS
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE IUTSAMEH
EZZ4308I ERROR: CODE=80103332 DURING ACTIVATION OF DEVICE ICPO3.
DIAGNOSTIC CODE: 02 5

EZZ4200I TCP/IP INITIALIZATION COMPLETE FOR T03ATCP
```

Figure 4-40 CS for z/OS IP startup

1 shows how the member that defines CTRACE processing has been found: CTIEZB01. We will discuss this member in 7.4, “Component trace (CTTRACE)” on page 158.

2 shows how the PROFILE.TCPIP for the CS for z/OS IP stack has been found and processed.

3 shows how the TCP/IP stack has been bound to UNIX System Services. It indicates that the Common INET pre-router has successfully obtained a copy of the IP layer routing table from the transport provider stack.

4 specifies the started task user ID of the transport provider stack for which a connection to UNIX System Services has been provided.

5 is interesting because it drives home the point that Communications Server for z/OS IP uses the data link control facilities of z/OS VTAM to support devices defined in the IP stack. This means that VTAM must be started prior to the start of CS for z/OS IP. The sense code in **5** is a VTAM DLC status code and must be interpreted with the help of *z/OS V1R2.0 CS: IP and SNA Codes*, SC31-8791. In this case it means that the device was not online.

Important: Since TCP/IP shares its Data Link Controls (DLCs) with VTAM, you must restart TCP/IP if you restart VTAM.



Multiple TCP/IP stacks on z/OS

Multiple TCP/IP stacks may coexist on the same z/OS system. If you are planning to run multiple copies of TCP/IP concurrently, you will need to understand issues that will help you define a strategy for doing so. In addition, you will need a background in the implementation and operational areas that affect your decision.

In this chapter we provide you with the information required to implement and configure a multiple-stack environment. We also give you console displays of multiple stacks in operation and provide you with examples of backing up and recovering using multiple stacks.

This chapter contains the following sections:

- ▶ 5.1, “Value of multiple concurrent copies of TCP/IP” on page 118
- ▶ 5.2, “Managing network attachments” on page 118
- ▶ 5.3, “Performance and capacity issues: multiple stacks” on page 120
- ▶ 5.4, “Common Internet Physical File System (CINET PFS)” on page 120
- ▶ 5.5, “Port management overview” on page 121
- ▶ 5.6, “SMF accounting issues: multiple stacks” on page 122
- ▶ 5.7, “Selecting a stack” on page 123
- ▶ 5.8, “Steps for installing a second stack” on page 129
- ▶ 5.9, “Example: implementing a two-stack configuration” on page 130

5.1 Value of multiple concurrent copies of TCP/IP

You can define as many as 32 Communications Server for z/OS IP stacks in the SYS1.PARMLIB(BPXPRMxx), and as many as eight stacks can be started at any one time.

Each protocol stack implementation has a separate IP address and host name and its own set of active, started interfaces. In fact, from a TCP/IP point of view, each protocol stack on the same z/OS system is a separate TCP/IP host system.

You might implement multiple stacks for any of a number of reasons:

- ▶ You might wish to establish separate stacks to separate workloads based on availability and security. For example, you might have different requirements for a production stack, a system test stack, and an education stack.


This approach could, for example, be used to establish a test TCP/IP stack, where new socket applications are tested before they are moved into the production system. The two TCP/IP address spaces can communicate with each other via, for example, a SAMEHOST link. You may want to apply maintenance to a non-production stack so it can be tested before you apply it to the production stack.

- ▶ Your strategy might be to separate workload onto multiple stacks based on the functional characteristics of applications, as with OpenEdition applications and non-OpenEdition applications.

In the past, there were other reasons to run multiple stacks, which have since been solved by added functionality:

- ▶ Running MVS servers and UNIX (OpenEdition) servers on the same well-known port (TN3270 and otelnet on port 23) can be easily overcome with the use of the BIND for INADDR_ANY function included with CS for OS/390 V2R10 IP. See Chapter 4, “Configuring base functions” on page 71 for more information.
- ▶ You might want servers with different configurations on the same well-known port (TN3270E and base TN3270 on port 23). Remote clients would use the host name (or address) of the appropriate stack to select the function they want to use. Again, the BIND for INADDR_ANY function included with CS for OS/390 V2R10 IP can overcome this restriction.

5.2 Managing network attachments

In order to save physical network interfaces, a design with a front-end stack that connects to the physical network and handles routing to application back-end stacks may be a desirable implementation. This design principle can be implemented purely with multiple stacks where one stack has the network interfaces and uses SAMEHOST links to route traffic to the back-end stacks. See diagram  in Figure 5-1.

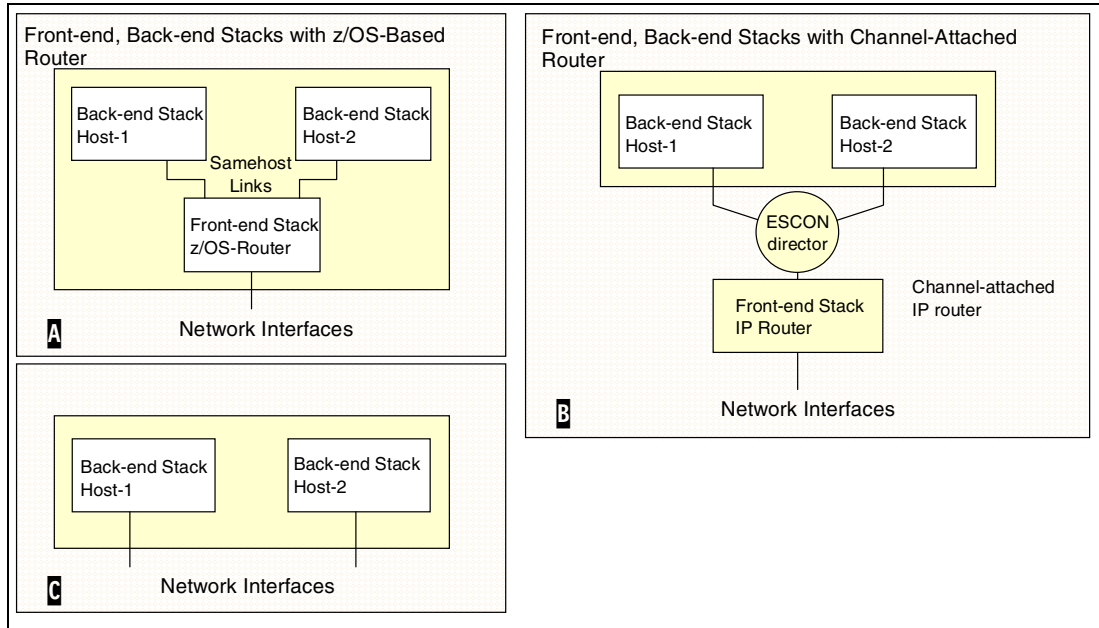


Figure 5-1 Managing network attachments

In a sysplex environment, it may be more desirable to offload the front-end stack functions to a channel-attached router that connects to the individual application back-end stacks via ESCON channel connections. Any channel-attached router can be used in such a configuration. See diagram **B** in Figure 5-1.

If you have only a couple of stacks, you may select to attach each of the stacks to the physical network via separate network connections. See diagram **C** in Figure 5-1.

5.2.1 Fault-tolerant network attachment

The front-end stack principle is good for saving physical network attachments, but it is also vulnerable if the front-end stack fails. A full implementation should therefore include backup for the front-end router, which can be accomplished by duplicating the channel-attached router and attaching each back-end stack to both channel-attached routers. If this technique is combined with the use of VIPA functions, TCP connections can be recovered dynamically using the alternate channel-attached router.

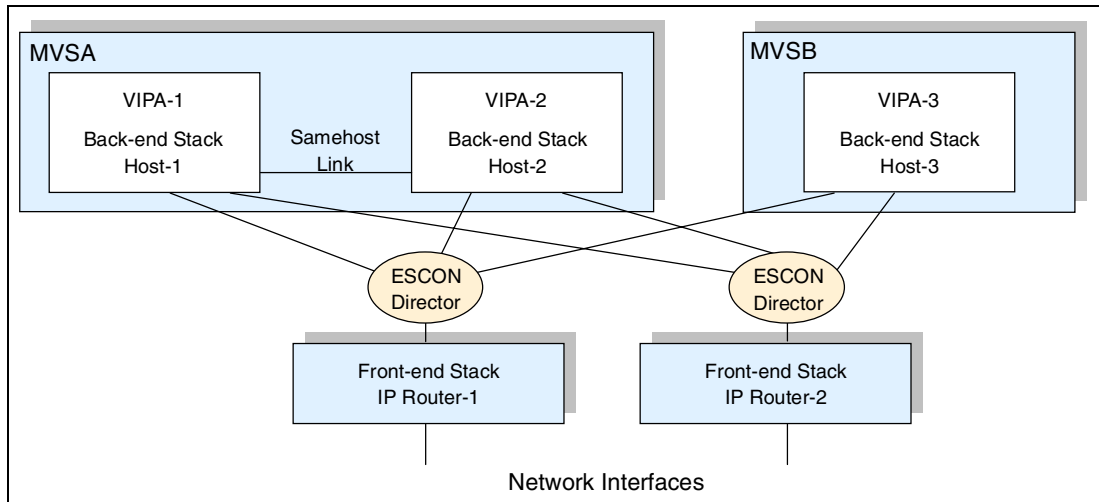


Figure 5-2 Fault-tolerant network attachment

This approach can also be used in an installation where the back-end stacks are executing on different z/OS systems, such as in a sysplex. The remote clients connect to one of the three VIPA addresses, and the IP traffic is routed via one of the two channel-attached routers.

5.3 Performance and capacity issues: multiple stacks

Consider the following trade-offs when designing an z/OS system that runs multiple stacks concurrently:

- ▶ If multiple stacks have LCS interfaces, CPU cycles spent on ARP processing will be duplicated by each stack, since each stack maintains its own ARP cache.
- ▶ If multiple stacks run servers that spend CPU cycles when certain periodic updates arrive, these CPU cycles will be duplicated in each stack. An example is that of routing daemons receiving periodic routing updates.
- ▶ Each stack requires a certain amount of system resources just to exist. The most significant resource is virtual storage. The amount varies depending on the configuration. Because of this, we do not recommend the use of multiple stacks except in isolated circumstances.

5.4 Common Internet Physical File System (CINET PFS)

In UNIX System Services, the Physical File System (PFS) includes the following components:

- ▶ Integrated Sockets AF-INET PFS
- ▶ Converged Sockets AF-INET, known as the Common Internet Physical File System (CINET PFS)
- ▶ Hierarchical File System (HFS)

The Integrated Sockets AF-INET PFS and the CINET PFS handle sockets requests from C programs and the UNIX System Services applications. The HFS PFS lets applications access files, then passes file requests from a UNIX System Services application, through DFSMS/MVS, to the HFS where traditional files or special character files are located.

Depending on the number of stacks you want to run on the sockets interfaces, you can use Integrated Sockets AF-INET or Common INET. Integrated Sockets AF-INET supports one TCP/IP stack at a time. It is used when applications communicate through a single stack. CINET is used when applications communicate through multiple stacks, and its daemon uses a timeout when opening sockets. For example, if you have three stacks and two of them are running, the inetd daemon opens the sockets on both of them. If the third stack is started later, the third stack cannot connect to any servers. To avoid this situation, start the servers after you start the stacks.

You can specify your choice of Integrated Sockets AF-INET or CINET in the NETWORK DOMAINNAME parameter of SYS1.PARMLIB(BPXPRMxx).

5.5 Port management overview

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

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

So, in a multiple transport provider environment, you need to solve some questions based on the following concepts:

► Generic server

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

Other examples of generic server/daemons shipped by CS for z/OS IP are:

- OE RSHD
- OE REXECD
- OE TELNETD
- TFTP
- DHCP
- TIMED
- OE Portmap

Note that the OE RSHD, OE REXECD and OE TELNETD are usually started by the inetd daemon that is shipped as part of the UNIX System Services. Since inetd is also a generic daemon, any server processes started by inetd inherently become a generic server as well.

► Servers with an affinity for a specific transport provider

There must be an explicit binding of the server application to the chosen transport provider when the service is related to the internal functioning of the transport provider. You can take UNIX System Services DNS, OROUTED, OSNMP and ONETSTAT as examples for that.

This bind is made via the setibmopt() socket call to specify which TCP/IP they have chosen, or via the C function _iptcpn() that allows applications to search in the TCPIP.DATA file to find the name of a specific TCP/IP.

- ▶ Ephemeral ports

As well as synchronizing PORT reservations for specific applications across all stacks, you have to synchronize reservations for port numbers that will be dynamically assigned across all stacks, when running with multiple transport providers.

Those ports are called ephemeral ports, which are all above 1024, and are assigned by the stack when none is specified on the application bind(). You have the PORTRANGE statement in the PROFILE.TCPIP to reserve a group of ports, and you should specify the same portrange for every stack. You also need to let CINET know which ports are guaranteed to be available on every stack, which is done in the BPXPRMxx parmlib member through INADDRANYPORT and INADDRANYCOUNT statements.

Taking those three concepts into consideration, you have to solve the following questions concerning port management:

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

5.6 SMF accounting issues: multiple stacks

Many installations rely on SMF for job accounting and for performance analysis. If you are running multiple stacks, SMF will not always allow you to distinguish among them. Consider the following issues:

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

FTP Server	One or six subtypes in FTP.DATA
Telnet Server	Two subtypes on TELNETPARMS
API	Two subtypes on SMFPARMS
FTP, Telnet Client	One subtype on SMFPARMS

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

5.7 Selecting a stack

Sockets application programs face two issues in a multi-stack environment:

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

In order to answer the above two questions, we need to distinguish between standard servers and clients (those that come with the Communications Server for z/OS product) and other sockets application programs, including those you might have written yourself.

5.7.1 Standard servers and clients

For standard servers and clients, the anchor configuration data set is the TCPIP.DATA data set. This is the main Resolver configuration data set with information on host name, domain origin, etc. In addition it also holds the TCPIPJOBNAME parameter, which identifies the TCP/IP stack to use, and it holds the DATASETPREFIX parameter, which is used by the Resolver code when allocating the other configuration data sets (HOSTS.SITEINFO, HOSTS.ADDRINFO, ETC.SERVICES, ETC.PROTO, and STANDARD.TCPXLBIN).

So the key both to selecting a specific stack and to selecting Resolver configuration data sets is to control which TCPIP.DATA data set is allocated by a standard server or client address space.

Non-OE servers and clients will search for TCPIP.DATA in the following sequence:

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

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

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

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

4. *userid*.TCPIP.DATA or *jobname*.TCPIP.DATA
5. SYS1.TCPPARMS(TCPDATA)
6. TCPIP.TCPIP.DATA

For the problem of determining which stack an application should use, an alternative solution is to try to have the well-known port be used by both the OE and MVS REXEC servers on the same MVS image. Obviously, this isn't possible on a single TCP/IP stack. A solution to this problem could be to configure two TCP/IP stacks on a single MVS image; one of the stacks would be designated as the OpenEdition Server stack and the other as the native MVS server stack. Note that in z/OS there really is no way to designate a TCP/IP stack as not being enabled for OpenEdition. Consider further that since a CINET environment is also required to execute multiple TCP/IP stacks, generic servers will typically be serviced by all stacks available. What's needed in this scenario is the ability to be able to bind the MVS REXEC

server to one stack and the OE REXEC server to the other. The MVS REXEC server always has affinity to the TCP/IP stack specified in the TCPIPJOBNAME parameter on its TCPIP.DATA file so this isn't a problem. However, as discussed earlier, the OE REXECD is started via inetd, which is a generic server. Therefore, in this scenario, we need to be able to have inetd, a generic daemon, have affinity to a specific stack.

This can be accomplished by use of the `_BPXK_SETIBMOPT_TRANSPORT` OpenEdition environment variable.

This environment variable, when set, has an effect similar to the `setibmopt()` function call provided by C/C++ compiler and described in the *z/OS V1R2.0 C/C++ Run-Time Library Reference*, SA22-7821. This variable can be set in the JCL for a started procedure or batch job that executes an OpenEdition C/C++ program to indicate which TCP/IP stack instance the application should bind to. OpenEdition TCP/IP applications that require affinity to a specific TCP/IP stack, such as OSNMPD and OROUTED, use the `setibmopt()` function call directly. The `_BPXK_SETIBMOPT_TRANSPORT` environment variable basically provides the ability to bind a generic server type of application to a specific stack.

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

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

```

    /*      for the EBCDIC-ASCII translation table. The file can
    /*      be any sequential data set, member of a partitioned data
    /*      set (PDS), or HFS file.
//SYSFTSX DD DISP=SHR,DSN=TCPV34.STANDARD.TCPXLBIN

```

Note that all the parameters specified prior to '/' in the parm statement are processed by the C/C++ runtime library. Parameters to be passed to the FTPD program must appear after the '/'. Also note how the parameters were split over three lines in this example, since they could not fit on a single line. Another example follows with JCL for the started procedure for inetd:

```

//INETD PROC
//*****
//INETD EXEC PGM=TO3INETD,REGION=OK,TIME=NOLIMIT,
//      PARM=(' POSIX(ON) ALL31(ON)',
//      'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE")',
//      '/ //' 'USER1.INETD.CONF' '')
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSERR DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*

```

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

On OS/390 V2R7 the inetd module has been moved to the Hierarchical File System. inetd has to be started with BPXBATCH.

```

//TO3INETD EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,
//      PARM='PGM /usr/sbin/inetd'
//STDENV DD DSN=TCP.TCPPARMS(IN3AENV),DISP=SHR

```

The //STDENV DD statement is needed to pick up the _BPXK_SETIBMOPT_TRANSPORT environmental variable. In this case the file should look like this:

```
BPXK_SETIBMOPT_TRANSPORT=TCPIPOE
```

The environmental variable _BPXK_SETIBMOPT_TRANSPORT will direct inetd to the TCPIPOE stack.

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

Notes:

- ▶ The _BPXK_SETIBMOPT_TRANSPORT variable is only supported when specified as a parm in JCL as previously described. It can be set via the STDENV DD file for programs started with BPXBATCH as shown in the above inetd example.
Use of it in other environments may yield unpredictable results.
- ▶ The _BPXK_SETIBMOPT_TRANSPORT variable should only be specified for a generic server type of application.
If specified for a non-generic server and/or non-UNIX (OpenEdition) C/C++ application, it will not have any effect.
- ▶ The name specified for _BPXK_SETIBMOPT_TRANSPORT must match the jobname associated with the TCP/IP stack.

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

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

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

```
EDC5112I Resource temporarily unavailable.
```

5.7.2 Non-standard servers and clients

Non-standard servers and clients also use TCPIP.DATA for deciding which Resolver configuration data sets to allocate, but they may or may not use the TCPIPJOBNAME parameter to select the stack. Whether or not they do depends on the socket API that was used to create the program.

If you run socket programs from other products or vendors, you may want to find out which socket API was used to develop the program and which techniques, if any, the program uses to specify the name of the TCP/IP system address space. As long as application programs that use a CS for z/OS socket library do not specify anything specific on a setibmopt() or INITAPI call, the TCPIPJOBNAME from a TCPIP.DATA data set will be used as the last resort for finding a TCP/IP system address space name.

The stack selection depends on the socket API you are running:

C Sockets	SETIBMOPT or TCPIPJOBNAME from TCPIP.DATA
Sockets Extended	TCPCNAME on INITAPI or TCPIPJOBNAME from TCPIP.DATA. Sockets Extended programs may have a configuration option to specify the TCP/IP system address space name, or they may interrogate the available stacks via the getibmopt() call. Sockets Extended programs do not have to call INITAPI. If INITAPI is not called, an implicit INITAPI will be performed with the value from TCPIPJOBNAME in a TCPIP.DATA data set. If INITAPI is called, a TCPCNAME of space results in the TCPIPJOBNAME keyword value being used as the TCP/IP system address space name.
Pascal Sockets	TCPIPJOBNAME from TCPIP.DATA
REXX Sockets	TCPIPJOBNAME from TCPIP.DATA

OpenEdition servers can use the setibmopt() function or the _BPXK_SETIBMOPT_TRANSPORT environment variable. OpenEdition clients can use the setibmopt() function or let the Common-INET pre-router select the stack depending on the destination IP address.

5.7.3 TCP/IP TSO clients

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

1. Add a SYSTCPD DD statement to your TSO logon procedure. The issue with this approach is that you will need a separate TSO logon procedure per stack you implement on your z/OS system, and users will have to log off TSO and log on again with another TSO logon procedure in order to switch from using one TCP/IP stack to another.

2. Use one common TSO logon procedure without a SYSTCPD DD statement. Before a TSO user starts any TCP/IP client programs, the user has to do a TSO **ALLOC** command where the user allocates a TCPIP.DATA data set to DD-name SYSTCPD. To switch from using one TCP/IP stack to another, the user simply has to deallocate the current SYSTCPD allocation and allocate another TCPIP.DATA data set.

The last method can be implemented easily by creating a small REXX program per TCP/IP stack on your z/OS system. For each stack, you create a REXX program with the name of the stack, for example, T03A or T03C. Whenever the TSO user wants to use the T03A stack, he/she runs the T03A REXX program. Any TCP/IP functions invoked thereafter will use the T03A stack for sockets communication. If the user wants to switch to the T03C stack, he or she runs the T03C REXX program. See Figure 5-3 for an example of such a REXX program.

```

/* REXX */
/*****/
/*
/* Switch TSO Address Space to use the T03A stack
/* Subsequent NETSTAT or PING commands will be directed towards
/* the T03ATCP stack.
/*
/*****/
Say 'Switching to T03ATCP stack'

msgstat = msg()
z = msg("OFF")
"FREE FI(SYSTCPD)"
"FREE FI(SYSFTPD)"
"ALLOC FI(SYSTCPD) DA('TCP.TCPPARMS(TDATA03A)') SHR"
z = msg(msgstat)

exit(0)

```

Figure 5-3 REXX program to switch TSO user to another TCP/IP stack

5.7.4 UNIX System Services clients

For UNIX System Services clients, it is typically the CINET pre-router that selects the stack depending on the destination IP address.

5.7.5 Selecting configuration data sets

The Resolver code that executes as part of the sockets program address space to service calls such as `gethostbyname()` or `getservbyname()` allocates one or more so-called Resolver configuration data sets in order to service these calls. All socket programs, including standard servers and clients and homegrown sockets programs, need access to one or more of these Resolver configuration data sets. In addition to TCPIP.DATA, the Resolver configuration data sets are HOSTS.ADDRINFO, HOSTS.SITEINFO, ETC.SERVICES, ETC.PROTO, and STANDARD.TCPXLBIN. Please refer to Figure 5-4.

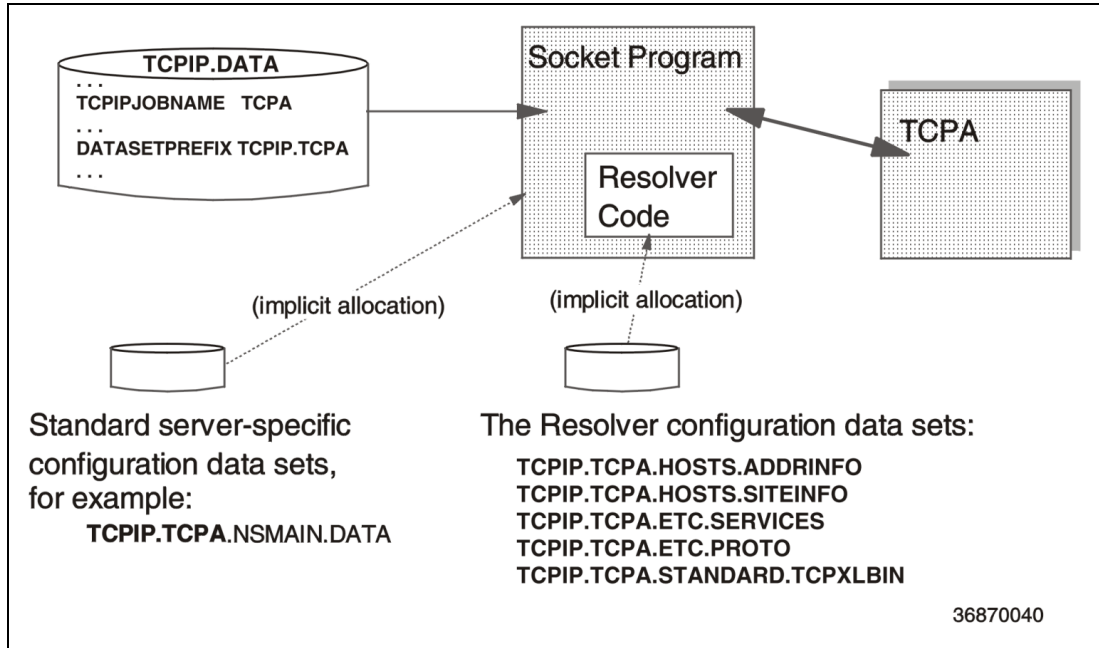


Figure 5-4 Selecting configuration data sets

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

5.7.6 Sharing Resolver configuration data sets between two stacks

The general recommendation is to use separate DATASETPREFIX values per stack and create separate copies of the required configuration data sets, or, at the very least, to create separate copies of the Resolver configuration data sets. For a test and a production stack, you would probably use different DATASETPREFIX values. However, if the stacks are functionally identical, you may share the same DATASETPREFIX value and many of the same configuration data sets. You need separate TCPIP.DATA data sets because of the two different TCPIPJOBNAMEs. On the other hand, you may choose to share the Resolver configuration data sets between the stacks by using the same DATASETPREFIX value in the two TCPIP.DATA data sets. Please refer to Figure 5-5 for an example.

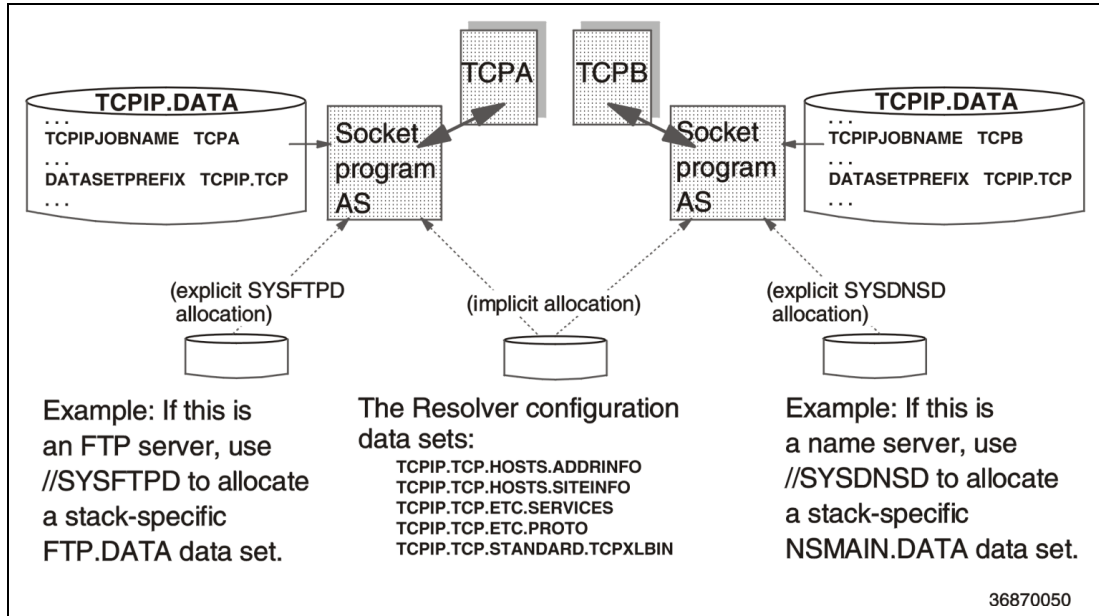


Figure 5-5 Sharing DATASET PREFIX

You must exercise caution if servers use the DATASET PREFIX to allocate server-specific configuration data sets. Try to use explicit allocation as far as possible in your server JCL procedures. Most servers allow you to explicitly allocate their configuration data sets via DD statements.

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

5.8 Steps for installing a second stack

The following steps apply to running two TCP/IP protocol stacks on the same z/OS system, with the same version and release level of TCP/IP.

1. Decide on a stack name and an associated stack DATASET PREFIX. For example, you might choose T03A as the stack name and TCPIP.T03A as the DATASET PREFIX.
2. Decide on network connections for the new stack. Remember that each stack is a separate TCP/IP host with its own network interface(s) and IP address(es). At a minimum a stack could have a SAMEHOST link to another stack on the same MVS system. A SAMEHOST link is a separate IP (sub)net with its own two endpoint IP addresses.
3. Decide whether you allocate a new TCPPARMS library to use for explicitly allocated configuration data sets for this stack, or create a new member in your existing TCPPARMS library, for example, TCP.TCPPARMS(TDATA03A) and TCP.TCPDATA(TDATA03C).
4. Alter your SYS1.PARMLIB(BPXPRMxx).
5. Create a PROFILE member in TCP.TCPPARMS.
6. Create a TCPDATA member in TCP.TCPPARMS.

7. Create a new system address space JCL procedure: T03CTCP.
8. Create required server address space JCL procedures with a SYSTCPD DD statement pointing to TCP.TCPPARMS(TDATA03C).
9. Create server-specific configuration data sets, such as FTP.DATA or NPF.DATA.

Note: If server-specific configuration data sets can be explicitly allocated using DD statements, we recommend that you create the configuration data set as a member in the stack-specific TCPPARMS library. If the data set has to be implicitly allocated, remember to create it with the stack-specific data set prefix.

10. Create required RACF definitions to assign started task user IDs to new address spaces.
11. If you are using a domain name server, ensure that it is updated with your new host name and address.
12. If you are not using a domain name server, edit your TCP.TCPPARMS(HOSTS) and run **MAKESITE**.
13. Optionally create a REXX program to switch TSO users to the new stack.

Note: An alternative to this is to create another TSO logon procedure and add //SYSTCPD pointing to the second TCPIP.DATA data set.

14. Depending on your system's management strategy, you may optionally create a different USS table and different VTAM definitions to distinguish among the different stacks.

5.9 Example: implementing a two-stack configuration

In this example we guide you through the creation of two TCP/IP stacks (T03A and T03C) on z/OS system MVS03 in the ITSO installation.

We perform the setup tasks following the steps in 5.8, "Steps for installing a second stack" on page 129.

5.9.1 Step 1: Stack name and DATASETPREFIX

The first stack will be called T03A. The TCP/IP system address space name will be T03ATCP and it will have a DATASETPREFIX of TCP.

The second stack will be called T03C. The TCP/IP system address space name will be T03CTCP and it will have a DATASETPREFIX of TCP as well.

5.9.2 Step 2: Network connections

See Figure 5-6 for an overview of the network configuration used by the two stacks.

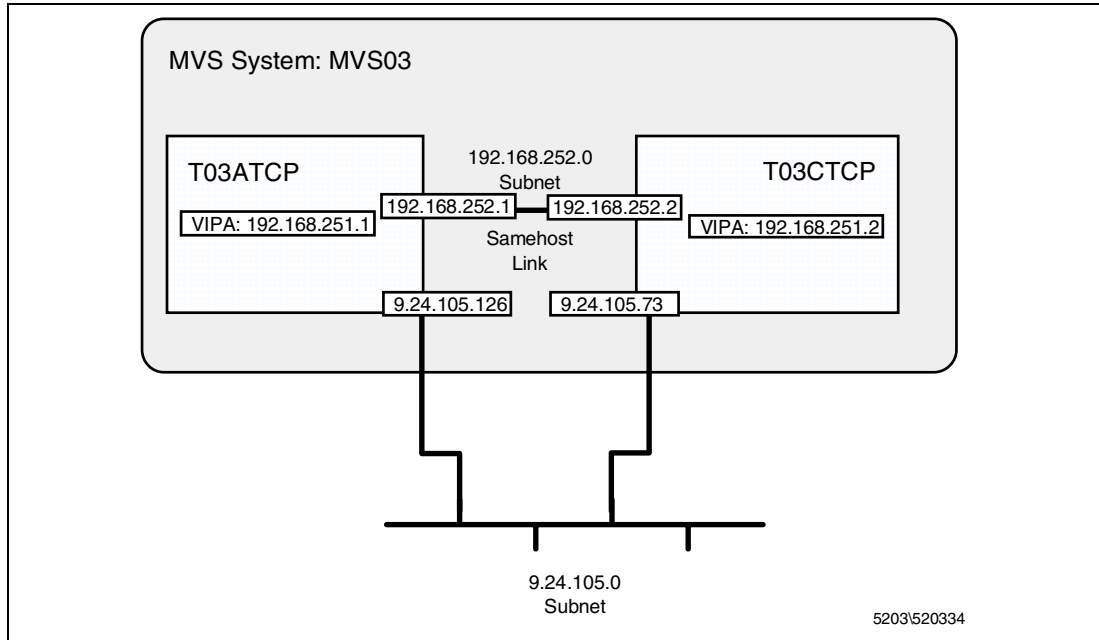


Figure 5-6 Two stacks on a z/OS system

The two stacks both have network interfaces to a token-ring. In addition to the LAN interfaces, they have a SAMEHOST link between them. Please note that the SAMEHOST link between the two stacks is a separate point-to-point link that requires a separate subnet.

In a configuration like the above, you have to be careful when you select the (sub)network address for the point-to-point link. If you are constrained on available subnets, it might be desirable to use a private network address on the SAMEHOST link, for example, a class C network of 192.168.252.0. To do so in the configuration shown would result in errors if you are using RIP-1 dynamic routing because you would introduce a discontinuous subnet situation, where subnets of the class A 9.0.0.0 network would be located at both ends of a class C network.

5.9.3 Step 3: Alter the BPXPRMxx member

You must have one SUBFILESYSTYPE entry for each stack, as in the following example for our configuration:

```

MAXPROCSYS(200)
MAXPROCUSER(25)
MAXUIDS(200)
MAXFILEPROC(64)
MAXPTYS(256)
CTRACE(CTIBPX00)
STEPLIBLIST('/SYSTEM/STEPLIB')
FILESYSTYPE TYPE(HFS)
        ENTRYPOINT(GFUAINIT)
FILESYSTYPE TYPE(IBMUDS)
        ENTRYPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
        DOMAINNUMBER(1)
        MAXSOCKETS(64)
        TYPE(IBMUDS)

FILESYSTYPE TYPE(CINET)

```

```

        ENTRYPOINT(BPXTICINT)
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(10000)
        TYPE(CINET)
        INADDRANYPORT(4000)
        INADDRANYCOUNT(2000)
SUBFILESYSTYPE NAME(T03ATCP)
        TYPE(CINET)
        ENTRYPOINT(EZBPFINI)
        DEFAULT
SUBFILESYSTYPE NAME(T03CTCP)
        TYPE(CINET)
        ENTRYPOINT(EZBPFINI)
ROOT FILESYSTEM('OMVS.SA03.ROOT.A')
        TYPE(HFS)
        MODE(RDWR)
MOUNT FILESYSTEM('OMVS.SA03.TMP')
        MOUNTPOINT('/tmp')
        TYPE(HFS)
        MODE(RDWR)
MOUNT FILESYSTEM('OMVS.SA03.USER')
        MOUNTPOINT('/u')
        TYPE(HFS)
        MODE(RDWR)
MOUNT FILESYSTEM('OMVS.SA03.ETC')
        MOUNTPOINT('/etc')
        TYPE(HFS)
        MODE(RDWR)
MOUNT FILESYSTEM('SMS.OMVS.KARL')
        TYPE(HFS)
        MODE(RDWR)
        MOUNTPOINT('/u/karl')
/* following are more user file systems. they are not shown here */

```

5.9.4 Step 4: Allocate TCPPARMS

We allocated only one data set TCP.TCPPARMS and used different members to hold the stack-specific configuration information.

5.9.5 Step 5: Create PROFILE.TCPIP

One member for each stack was created in the TCP.TCPPARMS library. Our examples will result in a TN3270 server running on stack T03A and OE Telnet (inetd) on stack T03C. Both Telnet servers can be reached at the well-known port 23.

Notice the AUTOLOG statements in the following examples. Stack T03A will autolog T03FTP and T03DNS, stack T03C will autolog inetd, and no stack will autolog WEBSRV. This configuration will need the following start sequence:

- ▶ Start the T03C stack.

The stack will autolog the inetd server. This server was defined as shown in 5.7.1, “Standard servers and clients” on page 123 to connect only to this stack.
- ▶ Start the T03A stack.

The stack will autolog T03FTP and T03DNS. The DNS server can only connect to one stack. The FTP server will connect to both stacks.
- ▶ WEBSRV was not defined in any autolog statements, so it has to be started manually.

Step 5A: profile for T03A stack

```

;*****
; Member TCP.TCPPARMS(PROF03A)
;*****
DATASETPREFIX TCP
TCPCONFIG
  UNRESTRICTLowports
  TCPSENDBfrsize 16384 ; Range is 256-256K - Default is 16K
  TCPRCVBufsize 16384 ; Range is 256-256K - Default is 16K
  SENDGARBAGE FALSE ; Packet contains no data
UDPCONFIG
  UNRESTRICTLowports
  UDPCHKsum ; Do checksum
  UDPSENDBfrsize 16384 ; Range is ???-???K (Default is 16K)
  UDPRCVBufsize 16384 ; Range is ???-???K (Default is 16K)
IPCONFig
  ARPTO 1200 ; In seconds
  DATAGRamfwd
  SOURCEVIPA
  VARSUBNETTING ; For RIPV2
  SYSPLEXRouting
  IGNORERedirect
  REASSEMBLYtimeout 15 ; In seconds
  STOPONclawerror
  TTL 60 ; In seconds, but actually Hop count
  AUTOLOG 5
  T03FTP JOBNAME T03FTP1 ; FTP Server
  T03DNS ; Domain Name Server
  WEBSRV ; Domain Name Server
  ENDAUTOLOG
PORT
  20 TCP OMVS NOAUTOLOG ; FTP Server
  21 TCP T03FTP1 ; FTP Server
  23 TCP INTCLIEN ; Telnet Server
  53 TCP T03DNS ; Domain Name Server - Parent Process
  53 UDP T03DNS ; Domain Name Server - Parent Process
  80 TCP OMVS ; Domino webserver
DEVICE DEVVIPA1 VIRTUAL 0
LINK LNKVIPA1 VIRTUAL 0 DEVVIPA1
DEVICE DEVEN1 LCS 2026
LINK EN1 802.3 1 DEVEN1
DEVICE IUTSAMEH MPCPTP
LINK T03CTCP MPCPTP IUTSAMEH
HOME
  192.168.251.1 LNKVIPA1 ; 1st VIPA Link (for V2R5)
  9.24.105.126 EN1 ; 9.24.105.0
  192.168.252.1 T03CTCP ; For SAMEHOST - IUTSAMEH - Connection
GATEWAY
  9 = EN1 1500 0.255.255.0 0.24.105.0
  192.168.252.2 = T03CTCP 4096 HOST
  DEFAULTNET 9.24.105.1 EN1 1500 0
;
; the Telnet parameters are in telnet3a
INCLUDE TCP.TCPPARMS(TELN03A)
;
START DEVEN1 ; OSA Ethernet
START IUTSAMEH ; SAMEHOST LINK (IUTSAMEH)

```

Step 5B: profile for T03C stack

```
*****
; Member TCP.TCPPARMS(PROF03C)
*****
DATASETPREFIX TCP
TCPCONFIG
  UNRESTRICTLowports
  TCPSENDBfrsize 16384 ; Range is 256-256K - Default is 16K
  TCPRCVBfrsize 16384 ; Range is 256-256K - Default is 16K
  SENDGARBAGE FALSE ; Packet contains no data
UDPCONFIG
  UNRESTRICTLowports
  UDPCHKsum ; Do checksum
  UDPSENDBfrsize 16384 ; Range is ???-???K (Default is 16K)
  UDPRCVBfrsize 16384 ; Range is ???-???K (Default is 16K)
IPCONFig
  ARPTO 1200 ; In seconds
  DATAGRamfwd
  SOURCEVIPA
  VARSUBNETTING ; For RIPV2
  SYSPLEXRouting
  IGNORERedirect
  REASSEMBLYtimeout 15 ; In seconds
  STOPONclawerror
  TTL 60 ; In seconds, but actually Hop count
  AUTOLOG 5
  INETD JOBNAME INETD1 ; INETD for OE telnet
  ENDAUTOLOG
PORT
  20 TCP OMVS ; FTP Server
  21 TCP T03FTP1 ; FTP Server
  23 TCP OMVS ; INETD for OE telnet
  80 TCP OMVS ; Domino webserver
  111 TCP OMVS ; Portmap Server
  111 UDP OMVS ; Portmap Server
DEVICE DEVVIPA1 VIRTUAL 0
LINK LNKVIPA1 VIRTUAL 0 DEVVIPA1
DEVICE DEVEN1 LCS 2026
LINK EN1 802.3 1 DEVEN1
DEVICE IUTSAMEH MPCPTP
LINK T03ATCP MPCPTP IUTSAMEH
HOME
  192.168.251.2 LNKVIPA1 ; 1st VIPA Link (for V2R5)
  9.24.105.73 EN1 ; 9.24.105.0
  192.168.252.2 T03ATCP ; For SAMEHOST - IUTSAMEH - Connection
GATEWAY
  9 = EN1 1500 0.255.255.0 0.24.105.0
  192.168.252.1 = T03ATCP 4096 HOST
  DEFAULTNET 9.24.105.1 EN1 1500 0
START DEVEN1 ; OSA Ethernet
START IUTSAMEH ; SAMEHOST LINK (IUTSAMEH)
```

5.9.6 Step 6: Create TCPIP.DATA

In the same way, one member was created for each stack in TCP.TCPPARMS.

Step 6A: TCPIP.DATA for the T03A stack

For the T03A stack, the contents are:

```

TCPIPJOBNAME T03ATCP
HOSTNAME MVS03
DOMAINORIGIN itso.ral.ibm.com
NSINTERADDR 9.24.104.108
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
;TRACE RESOLVER
DATASETPREFIX TCP
MESSAGECASE MIXED

```

Step 6B: TCPIP.DATA for the T03C stack

For the T03C stack, the contents are:

```

TCPIPJOBNAME T03CTCP
HOSTNAME MVS03C
DOMAINORIGIN itso.ral.ibm.com
NSINTERADDR 192.168.252.1 ; MVS03 Samehost Connection
NSINTERADDR 9.24.104.108
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 30
RESOLVERUDPRETRIES 1
DATASETPREFIX TCP
MESSAGECASE MIXED

```

5.9.7 Step 7: Create system address space JCL procedure

For the T03A stack, the system address name was selected to be T03ATCP:

```

//T03ATCP PROC PARM='CTRACE(CTIEZB01)',
// XPARM='ENVAR("RESOLVER_CONFIG=//TCP.TCPPARMS(TDATA03A)")'
//TCPIP EXEC PGM=EZBTCP,
// PARM='&PARMS &XPARM',
// REGION=7500K,TIME=1440
//STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//ALGPRINT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//SYSOUT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//SYSERROR DD SYSOUT=A
//PROFILE DD DISP=SHR,DSN=TCP.TCPPARMS(PROF03A)
//SYSTCPD DD DSN=TCP.TCPPARMS(TDATA03A),DISP=SHR

```

For the T03C stack, the system address space name is T03CTCP:

```

//T03CTCP PROC PARM='CTRACE(CTIEZB01)'
//TCPIP EXEC PGM=EZBTCP,
// PARM='&PARMS &XPARM',
// REGION=7500K,TIME=1440
//STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//ALGPRINT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//SYSOUT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=137,BLKSIZE=137)
//SYSERROR DD SYSOUT=A
//PROFILE DD DISP=SHR,DSN=TCP.TCPPARMS(PROF03C)
//SYSTCPD DD DSN=TCP.TCPPARMS(TDATA03C),DISP=SHR

```

5.9.8 Step 8: Create server address space JCL procedures

For each server you specified in the AUTOLOG section of the PROFILE configuration member, you need to create a JCL procedure in your JCL procedure library.

Taking Routed as an example, here is the JCL procedure for the T03AROUT server address space:

```
//T03AROUT PROC MODULE='BPXBATCH'  
//OROUTED EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,  
// PARM='PGM /usr/lpp/tcpip/sbin/orouted'  
//STDOUT DD PATH='/tmp/orouted.03a.stdout',  
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),  
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)  
//STDERR DD PATH='/tmp/orouted.03a.stderr',  
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),  
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)  
//STDENV DD PATH='/etc/orouted.03a.env'  
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)  
//SYSERR DD PATH='/tmp/orouted.03a.syserr',  
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),  
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
```

And this is the JCL procedure for the T03CROUT server address space:

```
//OROUTED EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,  
// PARM='PGM /usr/lpp/tcpip/sbin/orouted -ep -t -t -t'  
//* PARM='PGM /usr/lpp/tcpip/sbin/orouted'  
//STDOUT DD PATH='/tmp/orouted.03c.stdout',  
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),  
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)  
//STDENV DD DSN=TCP.TCPPARMS(RD03CENV),DISP=SHR
```

5.9.9 Step 9: Create server-specific configuration data sets

In this example, we need one configuration data set per stack, which can be explicitly allocated. As an example of the contents of those data sets, Examples 5-1 and 5-2 show the STDENV data set for each stack.

Example 5-1 STDENV data set for stack 03A (TCP.TCPPARMS(RD03A))

```
RESOLVER_CONFIG=/'TCP.TCPPARMS(TDATA03A)'  
ROUTED_PROFILE=/'TCP.TCPPARMS(RD03APR)'  
GATEWAYS_FILE=/'TCP.TCPPARMS(RD03AGW)'
```

Example 5-2 STDENV data set for stack 03C (TCP.TCPPARMS(RD03C))

```
RESOLVER_CONFIG=/'TCP.TCPPARMS(TDATA03C)'  
ROUTED_PROFILE=/'TCP.TCPPARMS(RD03CPR)'  
GATEWAYS_FILE=/'TCP.TCPPARMS(RD03CGW)'
```

5.9.10 Step 10: Update your name server

If you use a name server, you have to add the host names of the new stacks to your zone data sets. Remember that each stack is a separate TCP/IP host and has a separate host name with associated IP addresses.

5.9.11 Step 11: Create REXX program to switch TSO users

To switch a TSO user to use the T03A stack, we created the following REXX program, called T03A:

```
/* REXX */
/*****/
/*
/* Switch TSO Address Space to use the T03A stack
/* Subsequent client commands, such as NETSTAT, PING or FTP
/* will be directed towards the T03A stack.
/*
/*
/*****/
Say 'Switching to the T03A stack'

msgstat = msg()
z = msg("OFF")
"FREE FI(SYSTCPD)"
"FREE FI(SYSFTPD)"
"ALLOC FI(SYSTCPD) DA('TCP.TCPPARMS(TDATA03A)') SHR"
z = msg(msgstat)

exit(0)
```

To switch a TSO user to use the T03C stack, we created the following REXX program, called T03C:

```
/* REXX */
/*****/
/*
/* Switch TSO Address Space to use the T03C stack
/* Subsequent client commands, such as NETSTAT, PING or FTP
/* will be directed towards the T03C stack.
/*
/*
/*****/
Say 'Switching to the T03C stack'

msgstat = msg()
z = msg("OFF")
"FREE FI(SYSTCPD)"
"FREE FI(SYSFTPD)"
"ALLOC FI(SYSTCPD) DA('TCP.TCPPARMS(TDATA03C)') SHR"
z = msg(msgstat)

exit(0)
```

5.9.12 Step 12: Create VTAM definitions and USS message 10 tables

If both stacks are going to be used for Telnet access, you may wish to create two separate pools of VTAM LU definitions, one per stack.

The two stacks used the same USS message 10 table, but if you want to display the stack name on the USS message 10 screen, you need to create one USS message 10 table per stack.

5.9.13 Step 13: Starting the stacks

The two TCP/IP stacks are now ready to start.



National Language Support (NLS)

TCP/IP is an any-to-any communication protocol for heterogeneous systems. A general mechanism to negotiate the code page at the time of connection is not defined in the protocol suite. A client uses the services of a remote server. A server may service multiple clients at a given time residing on different systems and using different code pages. There is no way for the server to use the correct translation for each client. Some servers, however, allow the selection of a specific translate table by the client.

Communications Server for z/OS supports both single-byte character set (SBCS) and double-byte character set (DBCS) translation. For SBCS translation, one ASCII character translates to one EBCDIC character and vice versa. For DBCS translation, one ASCII double-byte character translates to one EBCDIC double-byte character and vice versa.

This chapter contains the following sections:

- ▶ 6.1, “Server and client translation options” on page 140
- ▶ 6.2, “Standard translate tables” on page 140
- ▶ 6.3, “Telnet use of translate tables” on page 143
- ▶ 6.4, “Code set conversion utilities in UNIX System Services” on page 145

For FTP support of multiple translate tables, please see *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

For LPD and LPR support of multiple translate tables, please see *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229.

z/OS UNIX Telnet relies on the z/OS UNIX **chcp** command to provide code page conversions. Therefore, if the Telnet client wants to use a code page other than the default IBM-1047, it has to use the **chcp** command. For more information, please see 6.4, “Code set conversion utilities in UNIX System Services” on page 145 and *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800.

6.1 Server and client translation options

The following TCP/IP server functions in z/OS offer a way for the remote client to select a specific translate table to be used by the z/OS server for a single session between the server and the client:

- ▶ The FTP server accepts the CTRLCONN, XLATE, and SBDAACONN keywords on the **site** subcommand for SBCS translation or the DBCS keywords on the **TYPE** subcommand for DBCS translation. The FTP server will optionally use the iconv functions instead of the external tables for single-byte conversion.
- ▶ The Telnet server, when it operates in DBCS transform mode, will prompt the user for a conversion type (which DBCS conversion to use).

The Telnet server relies on the UNIX System Services **chcp** command to provide code page conversion. The remote Telnet client can set the ASCII and EBCDIC code pages using the **chcp** command. For more information, refer to *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800.

The following TCP/IP client functions in CS for z/OS IP offer a way for the user to select a specific translate table to be used by the client for a single session between the client and the remote server:

- ▶ FTP via the TRANSLATE keyword on the **FTP** command for SBCS and/or DBCS translation, or the CTRLCONN and SBDAACONN keywords on the **locsite** subcommand for SBCS translation, or the DBCS client subcommands for DBCS translation.
- ▶ LPR via the translate keyword for SBCS translation or the DBCS keywords for DBCS translation.
- ▶ Telnet line mode connections via the translate keyword on the Telnet command invocation.
- ▶ TSO REXEC client via the **-t** parameter on the REXEC command invocation. You can use the SBCS translate table only.

There is always a default translate table selected during server or client startup, either through standard implicit allocation search order or through server configuration parameters. Please refer to *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 for the names of the default translate table data sets used by the individual server functions. During installation of Communications Server for z/OS IP, these default translate tables are created by the sample installation job EZAGETIN.

6.2 Standard translate tables

The different translate tables are supplied in editable source. A utility allows a user to convert the source into a binary file, which is actually used by TCP/IP. If you need a special translate table, you can create your own.

There are always two translate tables for each language, one for Telnet client usage and another for all the other applications.

The Telnet translate table prevents the z/OS Telnet client from translating special ASCII characters to EBCDIC characters, which are not allowed in the 3270 data stream and could lead to unpredictable results.

The standard translate tables are located in:

- ▶ SBCS tables: tcpip.SEZAXLD1

- DBCS tables: tcpip.SEZAXLD2

They are copied to sequential data sets by the sample installation job EZAGETIN.

6.2.1 Using your country SBCS translate table

Communications Server for z/OS IP contains the following SBCS NLS translate tables in source format.

Table 6-1 SBCS translations supported by CS for z/OS IP

Table name	Country or region	ASCII-EBCDIC code page numbers
AUSGER	Austrian-German	850<->273
BELGIAN	Belgian	850<->500
CANADIAN	Canadian	850<->037
CUSTOM	Customer	819<->1047
DANNOR	Danish-Norwegian	850<->277
DUTCH	Dutch	850<->037
FINSWED	Finnish-Swedish	850<->278
FRENCH	French	850<->297
ITALIAN	Italian	850<->280
JAPANESE	Japanese	850<->281
JPNALPHA	Japanese Code	1041<->1027
JPNKANA	Japanese Code	1041<->0290
KOR0891	Korean Code	0891<->0833
KOR1088	Korean Code	1088<->0833
PORTUGUE	Portuguese	850<->037
PRC1115	P.R. China	1115<->0836
SPANISH	Spanish	850<->284
SWISFREN	Swiss-French	850<->500
SWISGERM	Swiss-German	850<->500
TAI0904	Taiwan	0904<->0037
TAI1114	Taiwan	1114<->0037
UK	United Kingdom	850<->285
US	United States	850<->037

The editable translation tables used by the Telnet Server in 3270 DBCS transform mode and Telnet client are members of the tcpip.SEZATELX data set. The translation source tables used by the other applications, such as FTP, are located in the tcpip.SEZATCPX data set.

Note that the Telnet server for line mode no longer uses TCPXLBIN and hlq.TelnetSE.TCPXLBIN translation tables. Instead of them, now it uses iconv functions. You can specify the ASCII and EBCDIC code page with the CODEPAGE statement in the TelnetPARMS block.

All the translate table members contain two parts. The first part is used to translate from ASCII to EBCDIC. The second part is used to translate from EBCDIC to ASCII.

To use one of these translate tables, you must convert it into binary with the TSO **CONVXLAT** command.

The command **CONVXLAT 'tcpip.SEZATELX(FRENCH)' Telnet.TCPXLBIN** will create a translate table named *userid.Telnet.TCPXLBIN* available for the next Telnet line mode client command from this MVS user.

6.2.2 Using your country DBCS translate table

Communications Server for z/OS is delivered with the following DBCS translate tables in source format. The source for these tables is located in the tcpip.SEZADBCX data set.

Table 6-2 DBCS NLS translate tables

Member name	Language group	Translation	ASCII and EBCDIC code page numbers
EZACHLAT	Taiwan DBCS	Chinese	0927<->0835
		Big5	0947<->0835
EZAHLAT	Korea DBCS	Hanguel	0926<->0834
		KSC5601	0951<->0834
EZAKJLAT	Japan DBCS	EUckanji	0954<->0300
		JIS78kj	0955<->0300
		JIS83kj	5048<->0300
		SJiskanji	0301<->0300
EZASCLAT	P.R. China DBCS	Schinese	1380<->0837

These tables include only DBCS code points. To translate a mixed-mode data stream, you need to specify both an SBCS and a DBCS translation. A mixed-mode DBCS string is a string that contains both SBCS and DBCS characters. Shift-out and shift-in (EBCDIC X'0E' and X'0F') characters are used to denote the beginning and end of DBCS characters in a mixed-mode EBCDIC string.

You can modify the source of these tables and generate the corresponding binary translate tables with the CONVXLAT program.

The servers and clients that support DBCS translation are:

- ▶ FTP client and server - see *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.
- ▶ LPD client, LPD and SMTP server - see *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229.
- ▶ Telnet Server (for line mode) in 3270 DBCS transform mode.

6.3 Telnet use of translate tables

Telnet in line mode must always use its own translate tables in order to prevent the translation of special ASCII characters to EBCDIC characters outside the range of displayable data for the 3270 data stream.

6.3.1 Telnet sessions between two z/OS or VM hosts

Both sides of the Telnet functions, the server and the client function, have two modes:

- ▶ 3270 full-screen mode, where the data sent between the two TCP/IP hosts is an EBCDIC 3270 data stream. This is also called transparent mode.
- ▶ Line mode, where the data sent between the two TCP/IP hosts is ASCII.

3270 full-screen mode, Telnet server and client

No translation is done; the connection is transparent. Extended data stream including Programmed Symbols (PS) is supported.

Line mode, Telnet server

The Telnet server must translate from/to EBCDIC/ASCII because ASCII data is sent between the two TCP/IP hosts. The Telnet server for line mode no longer uses TCPXLBIN and hlq.TelnetSE.TCPXLBIN translation tables. Instead of them, now it uses iconv functions. You can specify the ASCII and EBCDIC code page with the CODEPAGE statement in the TelnetPARMS block.

Line mode, Telnet client

Users (Telnet clients) send and receive data to be displayed at their workstation in EBCDIC; therefore, EBCDIC/ASCII translation must be done on the client side. The user invoking the TSO Telnet Client function can request a specific translate table. The following hierarchical search algorithm is used during dynamic translate table data set allocation:

1. *user_id*.Telnet.TCPXLBIN
2. hlq.Telnet.TCPXLBIN
3. *user_id*.STANDARD.TCPXLBIN
4. hlq.STANDARD.TCPXLBIN

Only in line mode can you specify the TRANSLATE option to use a nonstandard translation table. If the TRANSLATE *dsname* option is specified, the Telnet client resolves the SBCS translation table in the following order:

1. *user_id.dsname*.TCPXLBIN
2. hlq.*dsname*.TCPXLBIN

If you specify this option, the STANDARD translation tables are never used. If *user_id.dsname*.TCPXLBIN and hlq.*dsname*.TCPXLBIN do not exist, or if they were incorrectly created, Telnet ends with an error message.

6.3.2 Telnet sessions between z/OS and other TCP/IP hosts

Both sides of CS for z/OS IP Telnet, the server and the client function, have three modes:

- ▶ 3270 full-screen mode

The data sent between the CS for z/OS IP host and the *other* TCP/IP host is an EBCDIC 3270 data stream.

- ▶ Line mode

The data sent between the CS for z/OS IP host and the *other* TCP/IP host is ASCII.

- ▶ DBCS transform mode

This mode supports full-screen access from the remote terminals that can emulate the family of DBCS-capable terminals, such as terminal types VT100, VT220, and VT282. The data sent between the CS for z/OS IP host and the *other* TCP/IP host is ASCII.

3270 full-screen mode, Telnet server

No translation is done; the connection is transparent. Extended data stream including programmed symbols (PS) is supported.

Note: The *other* TCP/IP host is responsible for building an EBCDIC 3270 data stream. In the case of an ASCII host, it must translate from EBCDIC to ASCII, which may include NLS translation.

This is known as *Telnet 3270 support* or *TN3270*.

If your workstation is an OS/2 workstation with IBM TCP/IP for OS/2, you can customize the translate table that TN3270 uses on the OS/2 workstation. It is located in the /tcpip/etc directory; the file name is 3278xlt. The format of this file matches the format of the translate table source members you find in tcpip.v3r1.SEZATELX. You can use FTP to download the translate table member that you want to use as 3278xlt.

Line mode, Telnet server

In this mode, the code conversion between EBCDIC and ASCII is done by the OS/390 Telnet server. The Telnet server uses iconv functions with the CODEPAGE statement in the TelnetPARMS block to specify country translation tables. You have to specify the ASCII and EBCDIC code sets using the code character names supported by the iconv functions.

You can allow a client to use the Telnet line mode server in MVS *without* any translation, if you specify the BINARYLINEMODE keyword on the TelnetPARMS statement in PROFILE.TCPIP.

DBCS transform mode, Telnet server

If you customize the MVS Telnet server to allow Telnet sessions in 3270 DBCS transform mode, a Telnet client will be prompted for the terminal type and which DBCS translation to use. In this mode, data exchanged between the client and the z/OS Telnet server is in ASCII, so all translation takes place in z/OS based on the selected DBCS translate table.

To enable 3270 DBCS transform mode, you have to include TNDBCSCN, TNDBCSSL and TNDBCSEER DD statements in your TCP/IP procedure.

Configure the TNDBCSCN data set with the CODEKIND and CHARMODE parameters according to the required DBCS code page. A sample is supplied in TCPIP.SEZAINST(TNDBCSCN).

The translate tables must reside in a partitioned data set allocated to the TCP/IP address space via a DD name of TNDBCSSL. The installation data set, hlq.SEZAXLD2, contains the default binary translation table. You can also customize the DBCS translation table for 3270 DBCS transform mode using the CONVXLAT utility.

TNDBCSEER DD receives trace output.

For more information, refer to *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

6.4 Code set conversion utilities in UNIX System Services

z/OS provides the `iconv` utility, which converts a file from one coded character set encoding to another. The `iconv` utility invokes `iconv()` functions internally such as `iconv_open()`, `iconv()`, and `iconv_close()`.

Several applications provided in Communications Server for z/OS IP use the functions to perform code conversion.

The following are examples that use the `iconv` functions to perform code conversion:

- ▶ The FTP server and client use `iconv` when `CTRLCONN` or `SBDATACONN` keywords are used in the `FTP.DATA` configuration file or on the `site` and `locsite` FTP subcommands to specify code pages. This support is limited to single-byte code pages. However, when transferring UCS-2 data, the EBCDIC code page can be double-byte.

- ▶ The UNIX System Services Telnet server

The Telnet server in a UNIX System Services (or TelnetD) environment provides the code conversion function based on the UNIX System Services `chcp` shell command. The Telnet client end user can specify which translation to use via the `chcp` shell command.

When a `chcp` shell command is executed, the TelnetD process is informed about the code page change via urgent data over the pseudo terminal connection (the master/slave PTY interface). A user may select to have a `chcp` command executed as part of the user's login process, for example, via the user's `$HOME/.profile` or `$HOME/.setup` shell scripts.

The code set conversions supported in CS for z/OS IP are shown in Table 6-3.

Table 6-3 Supported code character set converters

From code	To code
IBM-037	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-273	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-274	IBM-500, IBM-1047, ISO8859-1
IBM-275	IBM-500, IBM-1047, ISO8859-1
IBM-277	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-278	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-280	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-281	IBM-500, IBM-1047, ISO8859-1
IBM-282	IBM-500, IBM-1047, ISO8859-1
IBM-284	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-285	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-290	IBM-500, IBM-1027, IBM-1047, ISO8859-1
IBM-297	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-500	IBM-037, IBM-273, IBM-274, IBM-275, IBM-277, IBM-278, IBM-280, IBM-281, IBM-282, IBM-284, IBM-285, IBM-290, IBM-297, IBM-850, IBM-871, IBM-1027, IBM-1047, ISO8859-1
IBM-833	IBM-1047
IBM-836	IBM-1047

From code	To code
IBM-850	IBM-037, IBM-273, IBM-277, IBM-278, IBM-280, IBM-284, IBM-285, IBM-297, IBM-850, IBM-871, IBM-1047
IBM-871	IBM-500, IBM-850, IBM-1047, ISO8859-1
IBM-875	IBM-1047, ISO8859-7
IBM-930	IBM-1047
IBM-933	IBM-1047, ISO8859-1
IBM-935	IBM-1047
IBM-937	IBM-1047
IBM-939	IBM-1047
IBM-1026	IBM-1047, ISO8859-9
IBM-1027	IBM-290, IBM-500, IBM-1047, ISO8859-1
IBM-1047	IBM-037, IBM-273, IBM-274, IBM-275, IBM-277, IBM-278, IBM-280, IBM-281, IBM-282, IBM-284, IBM-285, IBM-290, IBM-297, IBM-500, IBM-833, IBM-836, IBM-850, IBM-871, IBM-875, IBM-930, IBM-933, IBM-935, IBM-937, IBM-939, IBM-1026, IBM-1027, ISO8859-1
ISO8859-1	IBM-037, IBM-273, IBM-274, IBM-275, IBM-277, IBM-278, IBM-280, IBM-281, IBM-282, IBM-284, IBM-285, IBM-290, IBM-297, IBM-500, IBM-871, IBM-933, IBM-1027, IBM-1047
ISO8859-7	IBM-875
ISO8859-9	IBM-1026

The DBCS code set converters are also supplied (see Table 6-4) in z/OS. These converters are used by the code set converters between the code sets IBM-930, IBM-932, IBM-932C, IBM-939, IBM-2022-JP, IBM-5052, IBM-eucJC, and IBM-eucJP.

Table 6-4 Supported DBCS code character set converters

From code	To code
IBM-290	IBM-932, IBM-932C, IBM-eucJP, IBM-eucJC
IBM-300	IBM-932, IBM-932C, IBM-eucJP, IBM-eucJC
IBM-930	IBM-932, IBM-932C, IBM-956, IBM-957, IBM-958, IBM-959, IBM-2202-JP, IBM-5052, IBM-5053, IBM-5054, IBM-5055, IBM-eucJP, IBM-eucJC
IBM-932	IBM-290, IBM-300, IBM-930, IBM-939, IBM-1027
IBM-932C	IBM-290, IBM-300, IBM-930, IBM-939, IBM-1027, IBM-1047
IBM-939	IBM-932, IBM-932C, IBM-956, IBM-957, IBM-958, IBM-959, IBM-1047, IBM-2202-JP, IBM-5052, IBM-5053, IBM-5054, IBM-5055, IBM-eucJP, IBM-eucJC
IBM-956	IBM-930, IBM-939
IBM-957	IBM-930, IBM-939
IBM-958	IBM-930, IBM-939
IBM-959	IBM-930, IBM-939

From code	To code
IBM-1027	IBM-932, IBM-932C, IBM-eucJP, IBM-eucJC
IBM-1047	IBM-930, IBM-939
IBM-2022-JP	IBM-930, IBM-939
IBM-5052	IBM-930, IBM-939
IBM-5053	IBM-930, IBM-939
IBM-5054	IBM-930, IBM-939
IBM-5055	IBM-930, IBM-939
IBM-eucJC	IBM-290, IBM-300, IBM-930, IBM-939, IBM-1027
IBM-eucJP	IBM-290, IBM-300, IBM-930, IBM-939, IBM-1027

Note: Specify IBM-932C or IBM-eucJC as the source or target code set name to set up for conversion of POSIX data encoded by IBM-932 or IBM-eucJP to or from a host code set encoding of the data such as IBM-930 or IBM-939.

Examples of POSIX data are C source and shell scripts. The data includes characters from the POSIX character set. The names IBM-932C and IBM-eucJC indicate that the <yen> and <overline> characters in POSIX data encoded by IBM-932 or IBM-eucJP map to the <backslash> and <tilde> characters, respectively, when the data is converted to or from host encodings.

You can use the name UCS-2 to request setup for conversion to and from the Universal Two-Octet Coded Character Set (UCS-2) specified in ISO/IEC International Standard 10646-1.

For example, if you specify IBM-939 as the EBCDIC code set, the code conversion to/from IBM-939 character encoding from/to UCS-2 character encoding is set up.

The EBCDIC code sets you can use for the UCS-2 conversion are shown in Example 6-1.

Example 6-1 EBCDIC code sets

IBM-037, IBM-273, IBM-274, IBM-275, IBM-277, IBM-278, IBM-280, IBM-282, IBM-284, IBM-285, IBM-290, IBM-297, IBM-300, IBM-420, IBM-424, IBM-500, IBM-813, IBM-819, IBM-833, IBM-834, IBM-835, IBM-836, IBM-837, IBM-838, IBM-850, IBM-852, IBM-856, IBM-861, IBM-862, IBM-864, IBM-866, IBM-869, IBM-870, IBM-871, IBM-880, IBM-904, IBM-912, IBM-914, IBM-915, IBM-916, IBM-920, IBM-921, IBM-922, IBM-927, IBM-930, IBM-932, IBM-933, IBM-935, IBM-937, IBM-939, IBM-942, IBM-946, IBM-948, IBM-949, IBM-950, IBM-951, IBM-964, IBM-970, IBM-1025, IBM-1026, IBM-1027, IBM-1046, IBM-1047, IBM-1088, IBM-1089, IBM-1112, IBM-1115, IBM-1122, IBM-1250, IBM-1251, IBM-1252, IBM-1253, IBM-1255, IBM-1256, IBM-1380, IBM-1381, IBM-1383, IBM-1386, IBM-1388, IBM-8550, IBM33722, IBM-eucJC, IBM-eucKR, IBM-eucTW, ISO8859-1, ISO8859-2, ISO8859-4, ISO8859-5, ISO8859-6, ISO8859-7, ISO8859-8, ISO8859-9

For more information about the iconv utility, please refer to *z/OS V1R2.0 C/C++ Programming Guide*, SC09-4765.

Diagnostic tools

z/OS V1R2.0 CS: IP Diagnosis, GC31-8782 is the ultimate source for diagnosing problems you may encounter in your implementation of Communications Server for z/OS IP. In this chapter we review some of the tools available to you, but you can find more detailed information in the referenced manual.

7.1 DISPLAY TCPIP command

You are already familiar with the MVS DISPLAY command as you have seen it in Chapter 2, “Customizing UNIX System Services” on page 17. The MVS **DISPLAY TCPIP** command displays the status of the TCP/IP stack or stacks.

You can retrieve a list of known stacks on an MVS system by issuing the MVS command **D TCPIP**:

```
D TCPIP
EZAOP50I TCPIP STATUS REPORT 012
COUNT  TCPIP NAME  VERSION  STATUS
-----  -
      1  TCPIPOE      CS V1R2  ACTIVE
      2  TCPIPMS      CS V1R2  ACTIVE
      3  TCPIPA       CS V1R2  ACTIVE
*** END TCPIP STATUS REPORT ***
```

The command is also useful in issuing other diagnostic commands, since you may need to distinguish among multiple running TCP/IP procedures. The **DISPLAY TCPIP** command gives you the procedure names.

7.2 NETSTAT and onetstat

NETSTAT, **onetstat**, and **DISPLAY TCPIP,tcpproc,NETSTAT** commands display information about the status of the local CS for z/OS IP configuration. They are extremely useful in diagnosing and monitoring your network. **onetstat** is issued from the UNIX System Services shell environment, **NETSTAT** is issued from TSO, and the **DISPLAY TCPIP,tcpproc,NETSTAT** command from the MVS console. All of them are functionally equivalent; they just use different front ends for input processing. They are all documented in *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781.

Any of the three variations of the command may be directed toward a specific stack if you are running multiple stacks, for example:

```

onetstat -p procname -d      1
D TCPIP,procname,N,DEV      2
TSO NETSTAT DEV TCP procname 3
  
```

- ▶ **onetstat** uses the **-p** option 1
- ▶ The **DISPLAY TCPIP,tcpproc,NETSTAT** command specifies the procedure name of the IP stack in the third field of the command 2
- ▶ The TSO **NETSTAT** command uses the TCP option 3

You can display online help by issuing the **onetstat -?**, **TSO NETSTAT HELP** or **DISPLAY TCPIP,tcpproc,HELP,NETSTAT** command.

Note: Access to the **NETSTAT** command can now be controlled by RACF. See *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781 or *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 for further details.

7.2.1 Routing table displays

To see information about each gateway use either the **onetstat -g** command (Figure 7-1) or the TSO **NETSTAT GATE** command.

```

onetstat -p TCPIPC -g
MVS TCP/IP onetstat CS V2R10      TCPIP Name: TCPIPC  14:53:04
Known gateways:
NetAddress  FirstHop      Link      Pkt Sz Subnet Mask Subnet Value
-----
Default    172.16.100.254 M032216B 30720 <none>
9.0.0.0    172.16.100.254 M032216B 30720 <none>
9.0.0.0    172.16.100.254 M032216B 30720 0.255.254.0 0.1.150.0
9.0.0.0    172.16.100.254 M032216B 30720 0.255.255.0 0.3.1.0
9.0.0.0    172.16.100.254 M032216B 30720 0.255.255.0 0.3.240.0
9.0.0.0    172.16.100.254 M032216B 30720 0.255.255.0 0.12.0.0
  
```

Figure 7-1 Gateway display

To display routing information, you can use the command **onetstat -r**, **TSO NETSTAT ROUTE** or the MVS console command **DISPLAY TCPIP,tcpproc,NETSTAT,ROUTE**.

```

D TCPIP,TCPIPC,NETSTAT,ROUTE
EZZ2500I NETSTAT CS V2R10 TCPIPC 875
DESTINATION      GATEWAY          FLAGS  REFCNT  INTERFACE
DEFAULT          172.16.100.254  UG     000000  M032216B
9.0.0.0          172.16.100.254  UG     000000  M032216B
9.1.150.0        172.16.100.254  UG     000000  M032216B
9.3.1.0          172.16.100.254  UG     000000  M032216B
9.3.240.0        172.16.100.254  UG     000000  M032216B
9.12.0.0         172.16.100.254  UG     000000  M032216B
9.12.2.0         172.16.100.254  UG     000000  M032216B
9.12.3.0         172.16.100.254  UG     000000  M032216B
9.24.105.0       172.16.100.254  UG     000000  M032216B
9.12.3.16        172.16.100.254  UG     000000  M032216B
9.12.3.32        172.16.100.254  UG     000000  M032216B
9.12.3.48        172.16.100.254  UG     000000  M032216B
9.12.0.0         172.16.100.254  UG     000000  M032216B
9.24.104.1       172.16.100.254  UGH    000000  M032216B
9.24.104.18      172.16.100.254  UGH    000000  M032216B

```

Figure 7-2 Displaying the route table

The FLAGS column is helpful to understand the characteristics of the route. Each flag represents certain information about the specific route entry, as follows:

- ▶ Flag U indicates that the route entry is up and running or ACTIVE. If there is no U, then the route entry is defined but not active. This may be because the device is in a NOT ACTIVE status.
- ▶ Flag G indicates that the route entry specifies an indirect route. That means the destination indicated on the route entry is behind a router from this z/OS system. If there is no G, then the route entry specifies a direct route. That means the destination indicated on the route entry is on the same local network.
- ▶ Flag H indicates that the destination field in this route entry specifies a host route. That means this route will be used only if the destination IP address of a datagram exactly matches all 32 bits (255.255.255.255) in the route entry destination field. If there is no H, then the destination field in this route entry specifies a network route. That means this route will be used only if the destination IP address of a datagram exactly matches all the network bits (less than 32 bits, for example, 255.255.255.0) in the route entry destination field.
- ▶ Flag D indicates that the route entry was created by an ICMP redirect. This may occur with static route definitions but not with dynamic routing protocols.
- ▶ Flag O indicates the route was created by OSPF.
- ▶ Flag R indicates the route was created by RIP.
- ▶ Flag S indicates the route is a static route that cannot be replaced by a routing daemon (such as OMPROUTE).
- ▶ Flag Z indicates the route is a static route that can be replaced by dynamic routes learned by OMPROUTE.

If you are using OMPROUTE, you can display routes in the OMPROUTE routing table by issuing the command **DISPLAY TCPIP,tcpproc,OMPROUTE,RTTABLE**.

7.2.2 Home addresses display

The **onetstat -h**, **TSO NETSTAT HOME** and **DISPLAY TCPIP,tcpproc,NETSTAT,HOME** commands display the home IP addresses for the IP stack. If you do not see your IP address in the home list, the interface will not be available for use.

```
D TCPIP,TCPIPC,NETSTAT,HOME
EZZ2500I NETSTAT CS V2R10 TCPIPC 775
HOME ADDRESS LIST:
ADDRESS          LINK              FLG
172.16.102.39    M392216B          P
172.16.233.39    EZASAMEMVS
172.16.233.39    EZAXCF28
172.16.233.39    EZAXCF03
172.16.251.3     VIPLAC10FB03      I
127.0.0.1        LOOPBACK
6 OF 6 RECORDS DISPLAYED
```

Figure 7-3 Home IP address table display

Flag P indicates the link VIPA39A is the primary interface.

Flag I indicates that this IP address was created as a result of this TCP/IP being identified as a target stack for this address from a sysplex distributing stack. This IP address is not advertised to routing daemons.

7.2.3 Device/link displays

The **onetstat -d** command displays the device status for the defined network interfaces. You can also use the **DEVLINKS** option in the **TSO NETSTAT** or **DISPLAY TCPIP,tcpproc,NETSTAT,DEVLINKS** command.

This command is used to verify whether a device or link is up and running. You can also issue the MVS command **DISPLAY U,,,device_addr,num** to check the status of the address assigned to that device. A state of **BSY/A** is normal. See *z/OS V1R2.0 MVS System Commands*, SA22-7627 for further information about the **DISPLAY U** command.


```

D TCPIP,TCPIPA,NETSTAT,DEVLINKS
EZZ2500I NETSTAT CS V1R2 TCPIPA 003
DEVNAME: LOOPBACK          DEVTYPE: LOOPBACK  DEVNUM: 0000
DEVSTATUS: READY
LNKNAME: LOOPBACK          LNKTYPE: LOOPBACK  LNKSTATUS: READY
NETNUM: 0  QUESIZE: 0
BYTESIN: 28659             BYTESOUT: 28659
BSD ROUTING PARAMETERS:
MTU SIZE: 00000           METRIC: 00
DESTADDR: 0.0.0.0         SUBNETMASK: 0.0.0.0
MULTICAST SPECIFIC:
MULTICAST CAPABILITY: NO
DEVNAME: OSA22E0          DEVTYPE: MPCIPA    DEVNUM: 0000
DEVSTATUS: READY          CFGROUTER: NON    ACTROUTER: NON
LNKNAME: OSA22EOLINK      LNKTYPE: IPAQENET LNKSTATUS: READY
NETNUM: 0  QUESIZE: 0  SPEED: 0000000100
BYTESIN: 12377            BYTESOUT: 11575
BROADCASTCAPABILITY: NO
ARPOFFLOAD: YES  ARPOFFLOADINFO: YES
BSD ROUTING PARAMETERS:
MTU SIZE: 00000           METRIC: 00
DESTADDR: 0.0.0.0         SUBNETMASK: 255.255.255.0
MULTICAST SPECIFIC:
MULTICAST CAPABILITY: YES
GROUP                     REFCNT
-----
224.0.0.1                 0000000001

```

Figure 7-4 Displaying devices and links

7.2.4 Active sockets displays

The `onetstat -c`, `onetstat -a`, and `onetstat -s` commands each display the current active socket connections. The `-a` option is basically the same as the `-c` option, but it displays information for all TCP/IP connections, including recently closed ones. The `-s` option also displays the subtask identifier. The subtask identifier is combined with the address space name to produce a unique identifier for the client.

The `-c`, `-a`, and `-s` options are equivalent to the `CONN`, `ALLCONN`, and `SOCKETS` options either for `TSO NETSTAT` or `DISPLAY TCPIP,tcpproc,NETSTAT` commands.

```

onetstat -p tcpipa -a
MVS TCP/IP onetstat CS V1R2          TCPIP Name: TCPIPA          18:14:31
User Id Conn      Local Socket          Foreign Socket              State
----- ----      -
TCPIPA  0000000B 127.0.0.1..1025      0.0.0.0..0                 Listen
TCPIPA  00000012 0.0.0.0..23001       0.0.0.0..0                 Listen
TCPIPA  00000013 0.0.0.0..23         0.0.0.0..0                 Listen
TCPIPA  00000011 0.0.0.0..23002       0.0.0.0..0                 Listen
TCPIPA  0000000E 127.0.0.1..1026      127.0.0.1..1025           Establish
TCPIPA  0000000F 127.0.0.1..1025      127.0.0.1..1026           Establish
TCPIPA  00000010 0.0.0.0..23003       0.0.0.0..0                 Listen

onetstat -p tcpipa -c
MVS TCP/IP onetstat CS V1R2          TCPIP Name: TCPIPA          18:16:01
User Id Conn      Local Socket          Foreign Socket              State
----- ----      -
TCPIPA  0000000B 127.0.0.1..1025      0.0.0.0..0                 Listen
TCPIPA  00000012 0.0.0.0..23001       0.0.0.0..0                 Listen
TCPIPA  00000013 0.0.0.0..23         0.0.0.0..0                 Listen
TCPIPA  00000011 0.0.0.0..23002       0.0.0.0..0                 Listen
TCPIPA  0000000E 127.0.0.1..1026      127.0.0.1..1025           Establish
TCPIPA  0000000F 127.0.0.1..1025      127.0.0.1..1026           Establish
TCPIPA  00000010 0.0.0.0..23003       0.0.0.0..0                 Listen
TCPIPA  00000048 0.0.0.0..1036        *.*                         UDP

onetstat -p tcpipa -s
MVS TCP/IP onetstat CS V1R2          TCPIP Name: TCPIPA          18:10:00
Sockets interface status:
Type  Bound to          Connected to          State  Conn
====  =====          =====
Name: TCPIPA  Subtask: 00000000
Stream 9.12.6.60..23001  9.24.106.91..1715   TimeWait 0000002E
Stream 9.12.6.60..23001  9.24.106.91..1714   TimeWait 0000002C
Stream 9.12.6.60..23001  9.24.106.91..1716   TimeWait 00000031
Name: TCPIPA  Subtask: 008CA0D8
Stream 0.0.0.0..23002  0.0.0.0..0          Listen 00000011
Name: TCPIPA  Subtask: 008CACF8
Stream 127.0.0.1..1026  127.0.0.1..1025     Establish 0000000E
Name: TCPIPA  Subtask: 008E1B58
Stream 127.0.0.1..1025  127.0.0.1..1026     Establish 0000000F
Stream 127.0.0.1..1025  0.0.0.0..0          Listen 0000000B
Name: TCPIPA  Subtask: 008E21A8
Stream 0.0.0.0..23003  0.0.0.0..0          Listen 00000010
Name: TCPIPA  Subtask: 008E2AE0
Stream 0.0.0.0..23     0.0.0.0..0          Listen 00000013
Name: TCPIPA  Subtask: 008E2C78
Stream 0.0.0.0..23001  0.0.0.0..0          Listen 00000012

```

Figure 7-5 Active sockets display

7.2.5 Connection detail display

The **onetstat -A** command provides detailed information about all TCP/IP connections. You can also use the ALL option in the **TSO NETSTAT** command. The following is an extract of the display representing one connection.

```

Client Name: TCPIPA                      Client Id: 00000012
Local Socket: 0.0.0.0..23001            Foreign Socket: 0.0.0.0..0
  Last Touched:      22:06:16           State:              Listen
  BytesIn:           0000000000         BytesOut:           0000000000
  SegmentsIn:        0000000000         SegmentsOut:        0000000000
  RcvNxt:            0000000000         SndNxt:            0000000000
  ClientRcvNxt:      0000000000         ClientSndNxt:      0000000000
  InitRcvSeqNum:     0000000000         InitSndSeqNum:     0000000000
  CongestionWindow: 0000000000         SlowStartThreshold: 0000000000
  IncomingWindowNum: 0000032768         OutgoingWindowNum: 0000000000
  SndWll:            0000000000         SndWl2:           0000000000
  SndWnd:            0000000000         MaxSndWnd:         0000000000
  SndUna:            0000000000         rtt_seq:           0000000000
  MaximumSegmentSize: 0000000536       OptMaxSegmentSize: 0000000000
  DSField:           00
  Round-trip information:
    Smooth trip time: 0.000             SmoothTripVariance: 1500.000
  ReXmt:             0000000000         ReXmtCount:        0000000000
  DupACKs:           0000000000
  SockOpt:           80                 TcpTimer:           00
  TcpSig:            00                 TcpSel:             00
  TcpDet:            C0                 TcpPol:             10
  PolicyRuleName:
  ReceiveBufferSize: 0000016384         SendBufferSize:    0000016384
  ConnectionsIn:     0000000014         ConnectionsDropped: 0000000000
  CurrentBacklog:    0000000000         MaximumBacklog:    0000000010

```

Figure 7-6 Detailed active sockets display

7.2.6 TCP/IP storage usage display

CS for z/OS V1R2 introduces a new command to display TCP/IP storage usage:

D TCPIP,tcpproc,STOR. Its output contains the following:

```

d tcpip,tcpipa,stor
EZZ8453I TCPIP STORAGE
EZZ8454I TCPIPA STORAGE CURRENT MAXIMUM LIMIT
EZZ8455I TCPIPA ECSA 3816K 4646K NOLIMIT
EZZ8455I TCPIPA POOL 4440K 4985K NOLIMIT
EZZ8459I DISPLAY TCPIP STOR COMPLETED SUCCESSFULLY

```

Figure 7-7 Displaying storage usage

7.2.7 NETSTAT filter enhancements

With CS for z/OS V1R2, you have the choice to include or exclude TN3270 server connections on the **netstat** commands listed below. The default is to include TN3270 server connections. To exclude them, specify option NOTN3270 on the **D TCPIP,tcpproc,NETSTAT** and **TSO NETSTAT** commands, and option **-T** on the **onetstat** commands.

► **TSO NETSTAT ALL**

onetstat -A

- ▶ **TSO NETSTAT ALLCONN**
 onetstat -a
 D TCPIP,tcpproc,NETSTAT,ALLCONN
- ▶ **TSO NETSTAT BYTEINFO**
 onetstat -b
 D TCPIP,tcpproc,NETSTAT,BYTEINFO
- ▶ **TSO NETSTAT CLIENTS**
 onetstat -e
- ▶ **TSO NETSTAT CONN**
 onetstat -c
 D TCPIP,tcpproc,NETSTAT,CONN
- ▶ **TSO NETSTAT SOCKETS**
 onetstat -s
 D TCPIP,tcpproc,NETSTAT,SOCKETS

You can now display sockets based on client name, IP address or port number. Use the **CLIENT**, **IPADDR** and **PORT** options on the **D TCPIP,tcpproc,NETSTAT,SOCKETS** and **TSO NETSTAT SOCKETS** commands, and the **-E**, **-I** and **-P** options on the **onetstat -s** command.

You can also specify options **IPADDR** and **PORT** on the **TSO NETSTAT ALL** command, and options **-I** and **-P** on the **onetstat -A** command, to select connections for a specific IP address or port number.

See *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781 for further details on these commands.

7.2.8 NETSTAT performance counters

In CS for z/OS V1R2, the commands **TSO NETSTAT ALL**, **onetstat -A**, **TSO NETSTAT DEVLINKS**, **onetstat -d** and **D TCPIP,tcpproc,NETSTAT DEVLINKS** have been updated to show performance characteristics. A new command **TSO NETSTAT STATS**, **onetstat -S**, **D TCPIP,tcpproc,NETSTAT,STATS** displays performance statistics.

For more information on these commands, see *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781. See *Communications Server for z/OS V1R2 Implementation Guide Volume 5 : Availability, Scalability and Performance*, SG24-6517 for information on CS for z/OS IP performance.

7.2.9 Monitoring Sysplex Distributor with NETSTAT

The following commands are available to help you monitor Sysplex Distributor activity:

1. Display information about Dynamic VIPA
 - **TSO NETSTAT VIPADYN**
 - **onetstat -v**
 - **D TCPIP,tcpproc,NETSTAT,VIPADYN**
2. Display Dynamic VIPA configuration data
 - **TSO NETSTAT VIPADCFG**
 - **onetstat -F**

- **D TCPIP,tcpproc,NETSTAT,VIPADCFG**
3. Display Dynamic VIPA connection routing table
 - **TSO NETSTAT VCRT**
 - **onetstat -V**
 - **D TCPIP,tcpproc,NETSTAT,VCRT**
 4. Display Dynamic VIPA destination port table
 - **TSO NETSTAT VDPT**
 - **onetstat -0**
 - **D TCPIP,tcpproc,NETSTAT,VDPT**
 5. Request Sysplex information
 - **D TCPIP,tcpproc,SYSPLEX**

Please refer to *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781 for more information on these commands.

7.3 PING/oping and TRACERTE/otracert commands

The **PING** and **TRACERTE** commands work as with earlier releases of TCP/IP. The **oping** and **otracert** commands are the UNIX shell versions of these commands. **PING** and **TRACERTE** are issued from TSO, **oping** and **otracert** from either the ISHELL or the UNIX System Services shell.

If you run multiple stacks, use option **-p tcpproc** on the **oping** command, and the **-a tcpproc** option on the **otracert** command to select the stack. TSO commands select the stack using the TCPIP.DATA data set.

The **PING** command is useful to check the network connectivity. With the **TRACERTE** command, you may find the last router the packet can reach in a disrupted network, or you can verify if the packets flow over a planned route.

```

oping -c 5 192.168.41.1

CS V2R10: Pinging host 192.168.41.1
Ping #1 response took 0.043 seconds.
Ping #2 response took 0.042 seconds.
Ping #3 response took 0.042 seconds.
Ping #4 response took 0.056 seconds.
Ping #5 response took 0.043 seconds.

otracert 192.168.41.1

Traceroute to 192.168.41.1 (192.168.41.1).
Use escape C sequence to interrupt
 1 172.16.100.254 (172.16.100.254)  6 ms  6 ms  6 ms
 2 P0K2210A.itso.ral.ibm.com (9.24.104.3)  9 ms  9 ms  8 ms
 3 192.168.41.1 (192.168.41.1)  40 ms  39 ms  39 ms

```

Figure 7-8 *oping* and *otracert*

With MULTIPATH support you can have multiple interfaces as possible paths for the same network destination. Option **-i** on the **oping** and **otracert** commands specifies the local interface over which the **oping** or **otracert** packets will be sent. There is no equivalent interface option on the **TSO PING** and **TSO TRACERTE** commands.

7.4 Component trace (CTRACE)

The MVS component trace is used for diagnosis of most TCP/IP problems. This includes the functions formerly associated with LESSTRACE, MORETRACE, PKTTRACE, and so on.

Note: Some components of TCP/IP continue to maintain their own tracing mechanisms as well, for example, the FTP server. Consult individual chapters in this redbook and others in the series for tracing and diagnosis methods unique to those components.

The following TCP/IP traces are available using the component trace:

- ▶ Event trace for TCP/IP stacks
- ▶ TCP/IP packet trace
- ▶ OMPROUTE trace
- ▶ Resolver trace
- ▶ TCP/IP intrusion detection service trace

Information APAR II12014 is a useful source of information on the TCP/IP component and packet trace.

For general information on the MVS component trace, see *z/OS V1R2.0 MVS Diagnosis: Tools and Service Aids*, GA22-7589.

7.4.1 Taking a component trace

Component trace data is written to either an external writer or the TCP/IP dataspace TCPIPDS1 (the default is to write trace data to the dataspace). The following command sequence starts a component trace that uses the external writer; this allows you to store trace data in data sets, which can later be used as input to IPCS.

In the following commands, component is one of the following:

- ▶ SYSTCPIP for the TCP/IP event trace
- ▶ SYSTCPDA for the TCP/IP packet trace
- ▶ SYSTCPRT for the OMPROUTE trace
- ▶ SYSTCPRE for the Resolver trace
- ▶ SYSTCPIS for the TCP/IP intrusion detection service trace

and proc_name is the name of the TCP/IP, OMPROUTE, or RESOLVER procedure.

1. Start the external writer (CTRACE writer):

```
TRACE CT,WTRSTART=ctwrt
```

Note: Before starting the external writer, ensure that you have the ctwrt procedure in the SYS1.PROCLIB library.

Figure 7-9 shows a sample of an external writer. It was taken from the IXZCTW member of SYS1.SAMPLIB. Other examples can be found in II12014 informational APAR.

```
//CTWDASD  PROC
//IEFPROC  EXEC  PGM=ITTTTCWR
//SYSPRINT DD   SYSOUT=A
//TRCOUT01 DD   DSN=&&TRACE,UNIT=3380,
//          SPACE=(4096,20),DISP=(NEW,PASS),DSORG=PS
```

Figure 7-9 External writer

2. Start the CTRACE and connect to the external writer:

```
TRACE CT,ON,COMP=component,SUB=(proc_name)
R xx,OPTION=(valid_options),WTR=ctwrt,END
```

3. Display the active component trace options with:

```
DISPLAY TRACE,COMP=component,SUB=(proc_name)
```

4. Perform the operation you want to trace.

5. Disconnect the external writer:

```
TRACE CT,ON,COMP=component,SUB=(proc_name)
R xx,WTR=DISCONNECT,END
```

6. Stop the component trace:

```
TRACE CT,OFF,COMP=component,SUB=(proc_name)
```

7. Stop the external writer:

```
TRACE CT,WTRSTOP=ctwrt
```

7.4.2 Event Trace for TCP/IP stacks (SYSTCPIP)

The TCP/IP event trace, SYSTCPIP, traces individual TCP/IP components, such as storage. It is automatically started at TCP/IP initialization using the default trace options set in the SYS1.PARMLIB member CTIEZB00. Alternatively, you may create a different member with new options (for example, CTIEZBXX) and override CTIEZB00 by means of the CTRACE keyword in your TCP/IP PROC:

```
//TCPIPC  PROC PARM='CTTRACE(CTIEZBXX) '
//*
//TCPIP   EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//        PARM='&PARMS'
```

Figure 7-10 Overriding CTIEZB00 with CTIEZBXX

If you wish to specify different trace options after TCP/IP initialization, you may execute the **TRACE CT MVS** command, and either specify a component trace options file or respond to prompts from the command. However, you might want to edit the buffer size in this member and increase its size from the default of 8 MB. Buffer sizes, unlike many of the other CTRACE options, cannot be reset without restarting the TCP/IP system address space.

Figure 7-11 shows the status of the component trace for TCP/IP procedure TCPIPC as it has been initialized using SYS1.PARMLIB member CTIEZB01. Note that we have changed the default value for BUFSIZE to 4M.

```

DISPLAY TRACE,COMP=SYSTCPIP,SUB=(TCPIPC)
IEE843I 10.31.34 TRACE DISPLAY 003
      SYSTEM STATUS INFORMATION
ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K)
TRACENAME
=====
SYSTCPIP
                                MODE BUFFER HEAD SUBS
                                =====
                                OFF          HEAD    3
      NO HEAD OPTIONS
SUBTRACE      MODE BUFFER HEAD SUBS
-----
TCPIPC        ON    0004M
ASIDS         *NONE*
JOBNAMES      *NONE*
OPTIONS       MINIMUM
WRITER        *NONE*

```

Figure 7-11 *DISPLAY TRACE,COMP=SYSTCPIP,SUB=(TCPIPC) output*

The MINIMUM trace option is always active. During minimum tracing, certain exceptional conditions are being traced so the trace records for these events will be available for easier debugging in case the TCP/IP system address space should encounter an abend condition.

Socket API trace

The SOCKAPI option for the TCP/IP CTRACE component SYSTCPIP is intended to be used for application programmers to debug problems in their application. The SOCKAPI option captures trace information related to the socket API calls that an application may issue. However, the SOCKET option is primarily intended for use by TCP/IP Service and provides information meant to be used to debug problems in the TCP/IP socket layer, UNIX System Services, or the TCP/IP stack. Please refer to *z/OS V1R2.0 CS: IP Diagnosis, GC31-8782* for further details on the SOCKAPI option.

7.4.3 Sample SYSTCPIP trace

Figure 7-12 shows the trace activation process (steps 1, 2 and 3) and Figure 7-13 shows the deactivation process (steps 5,6, and 7).

1. Start the external writer (CTRACE writer).
2. Connect to the CTRACE external writer and specify trace options.
3. Display the active component trace options.
4. Reproduce the failure you want to trace.
5. Disconnect the external writer.
6. Stop the component trace.
7. Stop the external writer.

Instead of specifying options individually as we did in Step 2, we could have created a SYS1.PARMLIB member that would have started the external writer for us and included the desired options. We could use the following command for that:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpproc),PARM=(CTIEZBXX)
```



```

TRACE CT,WTRSTART=CTVLAD
ITTO38I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K) 413
ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
IRR813I NO PROFILE WAS FOUND IN THE STARTED CLASS FOR 414
CTVLAD WITH JOBNAME CTVLAD. RACF WILL USE ICHRIN03.
IEF196I 1 //CTVLAD JOB MSGLEVEL=1
IEF196I 2 //STARTING EXEC CTVLAD
IEF196I STMT NO. MESSAGE
IEF196I 2 IEFC001I PROCEDURE CTVLAD WAS EXPANDED USING SYSTEM
IEF196I LIBRARY SYS1.PROCLIB
IEF196I 3 XXCTRACE PROC
IEF196I XX* EXTERNAL WRITER USED WITH CS V2R10
IEF196I 4 XXIEFPROC EXEC PGM=ITTRCWR
IEF196I 5 XXTRCOUT01 DD DSNAME=LUNA.CTRACE,DISP=OLD
IEF403I CTVLAD - STARTED - TIME=16.26.58
IEF196I IEF236I ALLOC. FOR CTVLAD CTVLAD
IEF196I IEF237I 0829 ALLOCATED TO TRCOUT01
IEF196I AHL906I THE OUTPUT BLOCK SIZE OF 27998 WILL BE USED FOR
IEF196I OUTPUT
IEF196I DATA SETS:
IEF196I LUNA.CTRACE
AHL906I THE OUTPUT BLOCK SIZE OF 27998 WILL BE USED FOR OUTPUT 441
DATA SETS:
LUNA.CTRACE
ITT110I INITIALIZATION OF CTRACE WRITER CTVLAD COMPLETE.
...
TRACE CT,ON,COMP=SYSTCPIP,SUB=(TCPIPC)
*90 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
R 90,JOBNAME=(TCPIPC),OPTIONS=(XCF)
IEE600I REPLY TO 90 IS;JOBNAME=(TCPIPC),OPTIONS=(XCF)
*91 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
R 91,WTR=CTVLAD,END
IEE600I REPLY TO 91 IS;WTR=CTVLAD,END
ITTO38I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K) 515
ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
...
DISPLAY TRACE,COMP=SYSTCPIP,SUB=(TCPIPC)
IEE843I 16.29.59 TRACE DISPLAY 526
SYSTEM STATUS INFORMATION
ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K)
TRACENAME
=====
SYSTCPIP
MODE BUFFER HEAD SUBS
=====
OFF HEAD 3
NO HEAD OPTIONS
SUBTRACE MODE BUFFER HEAD SUBS
-----
TCPIPC ON 0004M
ASIDS *NONE*
JOBNAMES TCPIPC
OPTIONS XCF
WRITER CTVLAD

```

Figure 7-12 CTRACE activation sample

```

TRACE CT,ON,COMP=SYSTCPIP,SUB=(TCPIPC)
*92 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
R 92,WTR=DISCONNECT
IEE600I REPLY TO 92 IS;WTR=DISCONNECT
*93 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
R 93,END
IEE600I REPLY TO 93 IS;END
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K) 597
ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
...

TRACE CT,OFF,COMP=SYSTCPIP,SUB=(TCPIPC)
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K) 613
ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
...

TRACE CT,WTRSTOP=CTVLAD
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
WERE SUCCESSFULLY EXECUTED.
IEF196I AHL904I THE FOLLOWING TRACE DATASETS CONTAIN TRACE DATA :
IEF196I LUNA.CTRACE
AHL904I THE FOLLOWING TRACE DATASETS CONTAIN TRACE DATA : 619
LUNA.CTRACE
IEE839I ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K) 623
ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
ITT111I CTRACE WRITER CTVLAD TERMINATED BECAUSE OF A WTRSTOP REQUEST.
..
-JOBNAME STEPNAME PROCSTEP RC EXCP CONN TCB SRB CLOCK
SERV PG PAGE SWAP VIO SWAPS
-CTVLAD IEFPROC 00 208 414 .00 .00 10.9
4286 0 0 0 0 0
IEF404I CTVLAD - ENDED - TIME=16.37.36
-CTVLAD ENDED. NAME- TOTAL TCB CPU TIME= .00
TOTAL ELAPSED TIME= 10.9
IEF196I IEF142I CTVLAD CTVLAD - STEP WAS EXECUTED - COND CODE 0000
...

DISPLAY TRACE,COMP=SYSTCPIP,SUB=(TCPIPC)
IEE843I 18.58.52 TRACE DISPLAY 384
SYSTEM STATUS INFORMATION
ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K)
TRACENAME
=====
SYSTCPIP
MODE BUFFER HEAD SUBS
=====
OFF HEAD 3
NO HEAD OPTIONS
SUBTRACE MODE BUFFER HEAD SUBS
-----
TCPIPC MIN 0004M
ASIDS *NONE*
JOBNAMES *NONE*
OPTIONS MINIMUM
WRITER *NONE*

```

Figure 7-13 CTRACE deactivation sample

7.4.4 Packet trace (SYSTCPDA)

Packet tracing captures IP packets as they enter or leave TCP/IP. You select what you want to trace via the PKTTRACE statement within the PROFILE.TCPIP or via the **VARY PKTTRACE** command entered from the MVS console. RACF authorization is required to execute this command.

With the **VARY PKTTRACE** command or PKTTRACE statement in PROFILE.TCPIP, you can specify options such as IP address, port number and protocol type. If you are planning to gather a trace for relatively long hours, or if your system experiences heavy traffic, it is recommended that you specify these filtering options, so that TCP/IP does not have to gather unnecessary packets.

The following steps run a packet trace, and the data is written to an external writer:

- a. Start the CTRACE external writer:

```
TRACE CT,WTRSTART=ctwtr
```

- b. Start the CTRACE and connect the external writer to the TCP/IP address space:

```
TRACE CT,ON,COMP=SYSTCPDA,SUB=(tcpprocname)  
R nn,WTR=ctwtr,END
```

- c. Check that the trace started successfully:

```
D TRACE,COMP=SYSTCPDA,SUB=(tcpprocname)
```

- d. Start the trace through the PROFILE.TCPIP statement and the **VARY OBEYFILE** command, or through the **V TCPIP, ,PKT** command.

```
VARY TCPIP,tcpprocname,PKT,ON
```

- e. Perform the operation that you want to trace.

- f. Stop the trace:

```
VARY TCPIP,tcpprocname,PKT,OFF
```

- g. Disconnect the external writer from TCP/IP:

```
TRACE CT,ON,COMP=SYSTCPDA,SUB=(tcpprocname)  
R nn,WTR=DISCONNECT,END
```

- h. Stop the CTRACE:

```
TRACE CT,OFF,COMP=SYSTCPDA,SUB=(tcpprocname)
```

- i. Stop the external writer.

```
TRACE CT,WTRSTOP=ctwtr
```

Examine the data with IPCS or send it to the Support Center if they have requested it.

Socket data trace

The data trace is used to trace socket data into and out of the Physical File System (PFS).

Follow the packet trace steps detailed above, but use the following commands to start and stop the data trace:

```
V TCPIP,tcpproc,DATTRACE,ON  
V TCPIP,tcpproc,DATTRACE,OFF
```

7.4.5 OMPROUTE trace (SYSTCPRT)

OS/390 V2R6 IP and later provides component trace support for the OMPROUTE application. A default minimum component trace is always started during OMPROUTE initialization. To customize the parameters used to initialize the trace, update the SYS1.PARMLIB member CTIORA00. Besides specifying the trace options, you can also change the OMPROUTE trace buffer size. The buffer size can be changed only at OMPROUTE initialization.

After OMPROUTE initialization, you must use the **TRACE CT** command to change the component trace options.

To gather the component trace for OMPROUTE, use the commands listed in 7.4.1, “Taking a component trace” on page 158, and specify a component name of SYSTCPRT and your OMPROUTE proc_name.

Examine the data with IPCS or send it to the Support Center if they have requested it.

7.4.6 Resolver trace (SYSTCPRE)

CS for z/OS V1R2 provides component trace support for the Resolver. A default minimum component trace is always started during Resolver initialization. To customize the parameters used to initialize the trace, update SYS1.PARMLIB member CTIRES00. Besides specifying the trace options, you can also change the Resolver trace buffer size. The buffer size can be changed only at Resolver initialization.

After Resolver initialization, you must use the **TRACE CT** command to change component trace options.

To gather the component trace for the Resolver, use the commands listed in 7.4.1, “Taking a component trace” on page 158, and specify a component name of SYSTCPRE and your Resolver proc_name.

Examine the data with IPCS or send to the Support Center if they have requested it.

7.4.7 Intrusion detection services trace (SYSTCPIS)

When the TCP/IP stack starts, it reads SYS1.PARMLIB member CTIIDS00, which contains trace options for the SYSTCPIS trace. Packets are traced based on IDS policy defined in LDAP.

Please see *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782 for details on the intrusion detection services trace, and *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775, for information on defining policy.

7.5 Obtaining component trace data with a dump

When dumping TCP/IP, remember to specify the dataspace name, which is always TCPIPDS1, since it contains the trace data for the SYSTCPIP, SYSTCPDA and SYSTCPIS components. Be sure to include “region” in the SDATA dump options:

1. DUMP COMM=(enter_dump_title_here)
2. Rxx,JOBNAME=tcpproc,DSPNAME=('tcpproc'.TCPIPDS1),CONT
3. Rxx,SDATA=(CSA,LSQA,NUC,PSA,RGN,SQA,SUM,SQA,TRT),END

To obtain a dump of the OMPROUTE or Resolver address space (which contain the trace table), use the **DUMP** command as follows:

1. **DUMP COMM=(enter_dump_title_here)**
2. **Rxx,JOBNAME=proc_name,SDATA=(RGN,CSA,SQA),END**

7.6 Analyzing a trace

You can format component trace records using IPCS panels or a combination of IPCS panels and the **CTRACE** command, either from a dump or from external writer files. You can also use IPCS in batch to print a component trace.

The primary purpose of the component trace is to capture data that the IBM Support Center may use in diagnosing problems. There is little information in the documentation on interpreting trace data. If you wish to analyze the packet trace or data trace by yourself, you can do so with IPCS, an example of the IP packet layout or format, and a booklet that allows you to interpret the hexadecimal data in EBCDIC or ASCII.

7.6.1 Using the IPCS panels

The code for the component trace formatter is in data set SYS1.MIGLIB. Ensure that your TSO logon procedure includes SYS1.MIGLIB in the STEPLIB concatenation.

```
----- IPCS PRIMARY OPTION MENU -----
OPTION ==> 0
                                *****
0  DEFAULTS   - Specify default dump and options      * USERID  - USER1
1  BROWSE    - Browse dump data set                  * DATE    - 98/03/26
2  ANALYSIS  - Analyze dump contents                 * JULIAN   - 98.085
3  UTILITY   - Perform utility functions             * TIME    - 16:14
4  INVENTORY - Inventory of problem data             * PREFIX  - USER1
5  SUBMIT    - Submit problem analysis job to batch  * TERMINAL- 3278
6  COMMAND   - Enter subcommand, CLIST or REXX exec  * PF KEYS - 12
T  TUTORIAL  - Learn how to use the IPCS dialog      *****
X  EXIT      - Terminate using log and list defaults

Enter END command to terminate IPCS dialog
Message Control ==> CONFIRM VERIFY FLAG(WARNING)
Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL

Press ENTER to update defaults.

Command ==>
```

Figure 7-14 IPCS main panel

First we specify the name of our trace data set on the IPCS defaults panel. See [Figure 7-15](#) on page 166.

```
----- IPCS Default Values ----- LOCAL

Command ==>

You may change any of the defaults listed below. The defaults shown
any changes are LOCAL. Change scope to GLOBAL to display global def

Scope ==> LOCAL (LOCAL, GLOBAL, or BOTH)

If you change the Source default, IPCS will display the current defa
Address Space for the new source and will ignore any data entered in
the Address Space field.

Source ==> DSNAME('TCP.CTRACE1.TRACE01') 1
Address Space ==> RBA
Message Routing ==> NOPRINT TERMINAL
Message Control ==> NOCONFIRM VERIFY FLAG(WARNING)
Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL

Press ENTER to update defaults.

Use the END command to exit without an update.
```

Figure 7-15 Changing IPCS defaults

After we return to the primary IPCS panel, we enter:

- ▶ 2 for ANALYSIS
- ▶ 7 for TRACES
- ▶ 1 for CTRACE

Once you arrive at the CTRACE primary option menu, you simply proceed through the menu selections, specifying Q first, then on the parameter panel displayed in Figure 7-16 on page 167 enter report type FULL and start formatting with s. See the trace shown in Figure 7-17 on page 167.

```

----- CTRACE QUERY PARAMETERS -----
COMMAND ==> S

Enter/verify CTRACE QUERY parameters below:

System      ==>          (System name or blank)
Component   ==>          (Blank for all active components)
Subnames    ==>

Report type ==> FULL    (Short or Full, short is default)

GMT/LOCAL   ==> G      (G or L, GMT is default)
Subname
entry panel ==>
Override
  source    ==>

CTRACE QUERY FULL

END/PF3 = return to CTRACE primary options panel.
S = start CTRACE. R = reset all fields.

```

Figure 7-16 Specifying query parameters and starting CTRACE formatting

```

IPCS OUTPUT STREAM ----- Line 0 C
Command ==>                                SCROLL =
***** TOP OF DATA *****

COMPONENT TRACE QUERY SUMMARY

      SYSNAME  COMPONENT  SUB NAME
      -----  -----
0001. MVSVIC97 SYSTCPDA TCPIPC 1

```

Figure 7-17 Displaying the trace formatting parameters

Now that you know the TCPIP name (TCPIPC 1 in Figure 7-17), you are prepared to execute the next step. Back out of CTRACE processing by pressing PF3 twice and select D to enter the formatting parameters on Figure 7-18 on page 168. Fill in the name of the TCP/IP proc 1 that was traced and that you want a FULL report 2

```

----- CTRACE DISPLAY PARAMETERS -----
COMMAND ==> S

System      ==>          (System name or blank)
Component   ==> SYSTCPDA 4 (Component name (required))
Subnames    ==> TCPIPC 1

GMT/LOCAL   ==> 1          (G or L, GMT is default)
Start time  ==>          (mm/dd/yy, hh:mm:ss.dddddd)
Stop time   ==>          (mm/dd/yy, hh.mm.ss.dddddd)
Limit       ==> 0          Exception ==>
Report type ==> FULL 2 (SHort, SUmmary, Full, Tally)
User exit   ==>          (Exit program name)
Override source ==>
Options     ==> 3

To enter/verify required values, type any character
Entry IDs ==> Jobnames ==> ASIDs ==> OPTIONS ==> SUBS ==

ENTER = update CTRACE definition.  END/PF3 = return to previous pane
S = start CTRACE.  R = reset all fields.

```

Figure 7-18 Specifying TCP/IP name and full report

In the Component field 4 specify:

- ▶ SYSTCPDA for packet trace or data trace
- ▶ SYSTCPIP for TCP/IP stack trace
- ▶ SYSTCPRT for OMPROUTE trace
- ▶ SYSTCPRE for the Resolver trace
- ▶ SYSTCPIS for intrusion detection services trace

In Options 3 specify DATATRACE for a data trace, or PACKETTRACE for a packet trace. You can use the Options field to filter the display: for example, if you have a packet trace and you only want to display packets for port 23, you can specify PACKETTRACE PORT(23), or if you have a TCP/IP event trace and you only want to display events related to a particular IP address you can specify IPADDR(ip_addr). See *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782 for a full list of available options.

In CS for z/OS V1R2, the CTRACE packet trace formatter has been rewritten. The example below shows one packet which has been formatted using the new packet trace formatter.


```

COMPONENT TRACE FULL FORMAT
SYSNAME(SC64)
COMP(SYSTCPDA)SUBNAME((TCPIPA))
OPTIONS((PACKETTRACE))
OS/390 TCP/IP Packet Trace Formatter, (C) IBM 2000, 2001.134
DSNAME('WOODS.CTRACE1')

**** 2001/09/24
RcdNr Sysname Mnemonic Entry Id Time Stamp Description
-----
40 SC64 PACKET 00000001 17:55:02.444686 Packet Trace
To Link : OSA22EOLINK Device: QDIO Ethernet Full=71
Tod Clock : 2001/09/24 17:55:02.444686
Lost Records : 0 Flags: Pkt Ver2 Adj Out
Source Port : 23 Dest Port: 1160 Asid: 004C TCB: 00000000
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 71 ID Number: 050C
Fragment : Offset: 0
TTL : 64 Protocol: TCP CheckSum: F34F FFFF
Source : 9.12.6.60
Destination : 9.24.105.246

TCP
Source Port : 23 (telnet) Destination Port: 1160 ( )
Sequence Number : 2366825366 Ack Number: 2895763969
Header Length : 20 Flags: Ack Psh
Window Size : 32765 CheckSum: B735 FFFF Urgent Data Pointer: 0000

Telnet: 31
0000 IAC,WILL,SUPPRESS GO AHEAD,(data=28);

IP Header : 20 IP: 9.12.6.60, 9.24.105.246
000000 45000047 050C0000 4006F34F 090C063C 091869F6
Protocol Header : 20 Port: 23, 1160
000000 00170488 8D12E396 AC99DA01 50187FFD B7350000

Data : 31 Data Length: 31
000000 FFFB030D 0A494B4A 35363730 30412045 |.....¢..... .....IKJ56700A E|
000010 4E544552 20555345 52494420 3D0D0A |+..... NTER USERID =.. |

```

Figure 7-19 Formatted packet

7.6.2 Using IPCS and the CTRACE command

If you prefer to bypass most of the IPCS panels, you can enter IPCS, set your defaults (Option 0), and go to command mode (Option 6). Then use the **CTTRACE** command to format the trace data. For example:

```

CTTRACE COMP(component) SUB((proc_name)) FULL
CTTRACE COMP(SYSTCPDA) SUB((TCPIPA)) FULL OPTIONS((PACKETTRACE PORT(23)))

```

For more information on the **CTTRACE** command see *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782.

7.6.3 Printing a component trace

If you want to print a component trace, you may do so with IPCS. See Figure 7-20 on page 170 for a sample batch IPCS job to format a TCP/IP component trace.

```
//jobname JOB ...
//IEFPROC EXEC PGM=IKJEFT01,REGION=4M,DYNAMNBR=10
//STEPLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//IPCSPARM DD DSN=SYS1.PARMLIB,DISP=SHR
//IPCSDDIR DD DSN=userid.DDIR,DISP=SHR
//SYSPROC DD DSN=SYS1.SBLSCLIO,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//IPCSTOC DD SYSOUT=*
//IPCSPRNT DD SYSOUT=*
//SYSTSIN DD *
IPCS
MERGE
CTRACE DSN('your.trace.dataset1') -
      COMP(component) SUB(procname) FULL -
      OPTIONS((option1,option2))
CTRACE DSN('your.trace.dataset2') -
      COMP(component) SUB(procname) FULL -
      OPTIONS((option1,option2))
MERGEEND
END
/*
```

Figure 7-20 Batch IPCS formatting job

The component and procname are the same values you specified when you took the component trace. The options vary depending on the type of component trace. Please refer to *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782 for details of the options available for each trace. IPCS requires member IPCSPR00 to be present in the data set referenced by DD name IPCSPARM (usually SYS1.PARMLIB). See *z/OS V1R2.0 MVS Initialization and Tuning Reference*, SA22-7592 for details of what to define in IPCSPR00.

7.6.4 Useful formats

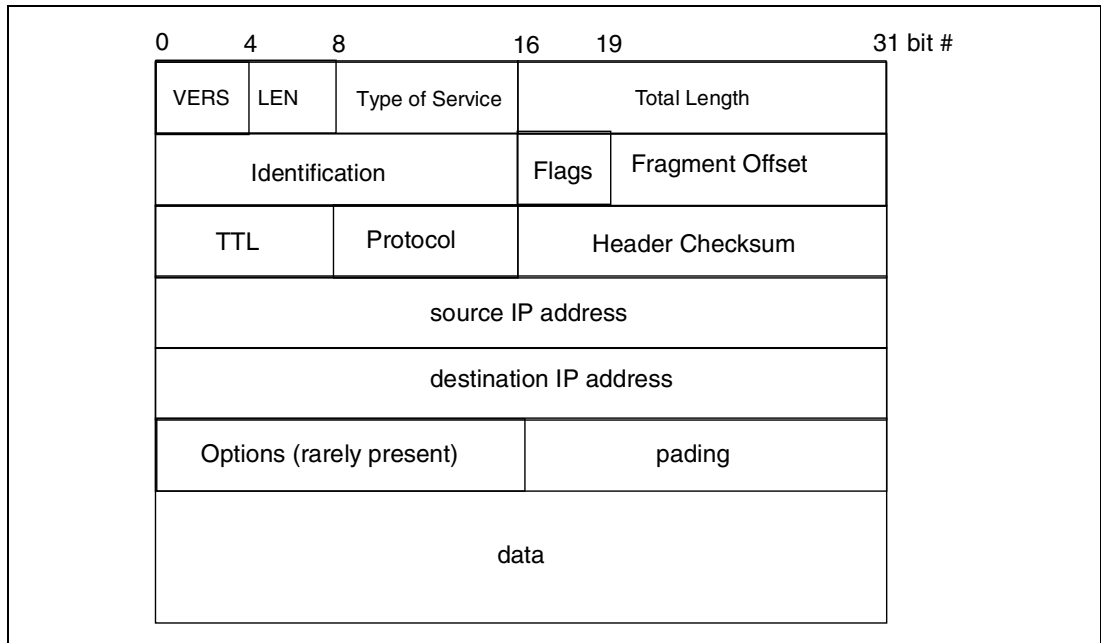


Figure 7-21 Format of IP datagram

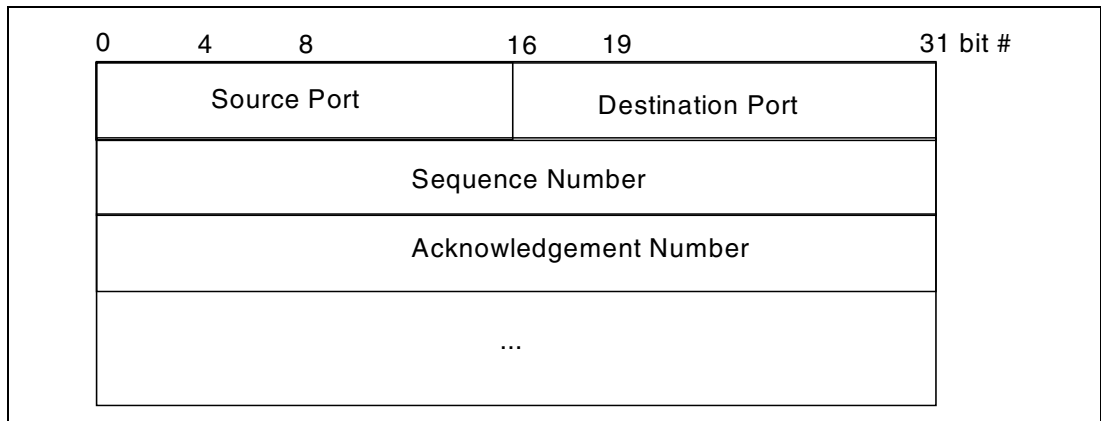


Figure 7-22 Format of TCP header

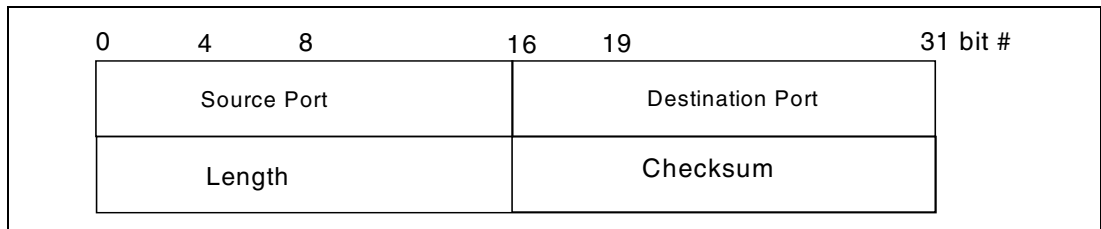


Figure 7-23 Format of UDP header

7.7 Processing IPCS dumps

Communications Server for z/OS IP provides an IPCS subcommand to format an IPCS dump. The TCPIPCS subcommand is documented in *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782.

7.8 Configuration profile trace

You can use the ITRACE statement in the PROFILE.TCPIP data set to activate TCP/IP runtime tracing for configuration, the TCP/IP SNMP subagent, commands, and the autolog subtask. ITRACE should only be set at the direction of an IBM Service representative. For more information, please refer to *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782.

7.9 Job log versus syslog as diagnosis tool

In the past, we often used the TCP/IP job log to detect problems. Most procedures now send messages to the syslog daemon or the MVS console log. Please refer to the *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228 for more information on the syslog daemon. Individual server documentation also provides information about diagnosis.

7.10 Message types: where to find them

You will want to have the z/OS system messages manuals available, as well as the messages manuals for TCP/IP. You will need to understand messages with a BPX prefix, an EZn prefix, and with SNA sense codes.

- Messages with prefix of BPX
You will find the explanations for these messages in *z/OS V1R2.0 MVS System Messages, Vol 3 (ASB-BPX)*, SA22-7633.
- Messages with prefix of EZA
For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 1 (EZA)*, SC31-8783.
- Messages with prefix of EZB
For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 2 (EZB)*, SC31-8784.
- Messages with prefix of EZY
For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 3 (EZY)*, SC31-8785.
- Messages with prefix of EZZ and SNM
For Communications Server for z/OS IP, you will find the explanations for these messages in *z/OS V1R2.0 CS: IP Messages Volume 4 (EZZ-SNM)*, SC31-8786.
- Messages with prefix of FOMC, FOMM, FOMO, FSUC and FSUM
You will find the explanations for these messages in *z/OS V1R2.0 UNIX System Services Messages and Codes*, SA22-7807.

- Eight-digit SNA sense codes and DLC codes
You will find the explanations for these codes in *z/OS V1R2.0 CS: IP and SNA Codes*, SC31-8791.
- UNIX System Services return codes and reason codes
You will find the explanations for these codes in *z/OS V1R2.0 UNIX System Services Messages and Codes*, SA22-7807.



TN3270 Telnet server

This chapter focuses on the Telnet functions that are available in z/OS V1R2 Communications Server IP. Where appropriate, we indicate the level of CS for z/OS IP at which a specific function was introduced.

8.1 Overview

Telnet is a terminal emulation protocol that allows you to log on to a remote host as if you were directly connected to it. Telnet allows users to have access to applications running on that host. You may establish concurrent sessions on different hosts or multiple sessions with a single host.

Telnet operation

Telnet has a concept of Network Virtual Terminal (NVT). NVT is a virtual device that has basic characteristics common to a wide range of real terminals. NVT must be supported by all Telnet servers and clients.

The Telnet protocol allows servers and clients to negotiate their characteristics, because many hosts will wish to provide additional services to the NVT. Once a TCP connection has been established, both sides of the connection are capable of working at the minimum level that is implemented by NVT. After this minimum understanding is achieved, they can negotiate additional options to extend the capabilities of the real hardware in use. Because of the symmetric model used by Telnet, both the server and the client may propose additional options to be used.

Telnet 3270 (TN3270)

The TN3270 or TN3270E Telnet server is simply a VTAM application that activates one application minor node logical unit (LU) to represent each Telnet client. You should note that the Telnet server is a separate application, not a part of VTAM, although it uses VTAM. Using TN3270 Telnet is a two-step process. First the client connects via TCP/IP to the Telnet server listening on some TCP port. The Telnet server allocates VTAM resources and a session on behalf of the client.

Traditional Telnet may be used to make a TCP/IP connection to an SNA host, but Telnet 3270 will be the better choice to connect to the host because it provides 3270 emulation capability. This capability will release the host from the responsibility of converting EBCDIC to/from ASCII code, and avoids the imperfection of the code conversion. TN3270 server does not convert the SNA data stream.

The following differences between traditional Telnet and 3270 terminal emulation make it necessary to use 3270 emulation under certain circumstances:

- ▶ 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 TN3270 connection is accomplished by the negotiation of the following Telnet options:

- ▶ Terminal Type
- ▶ Binary Transmission
- ▶ End of Record

The Terminal Type option is a string that specifies the terminal type for the host such as IBM-3278-2-E. The Binary Transmission option specifies that the receiver should interpret characters received from the sender as 8-bit binary data, besides the "interpret as command" (IAC) character and the following Telnet command. Since the length of the data may vary, and CRLF no longer means "new line" in binary transmission mode, every command and its related data must be separated with the IAC EOR sequence. For this purpose, the End of Record option is used.

Note: TN3270E is a function and IBM 327x device types that end in -E (for example, 3278-2-E) are terminals that support Extended field attributes such as color and highlighting. These field attributes are not related to Telnet functions.

TN3270 enhancements (TN3270E)

The TN3270 function was enhanced by RFC 1646 and RFC 1647. Since RFC 1647 covers all of the function that RFC 1646 has and most clients support RFC 1647, z/OS Telnet does not implement RFC 1646.

Note: RFC 1646 and RFC 1647 are *not* compatible. If your TN3270 client supports only RFC 1646, you will not have LU name selection and printer support capabilities.

The z/OS Telnet server implements RFC 1647 TN3270 enhancements. TN3270E overcomes the following shortcomings in traditional TN3270:

- ▶ It provides no capability to emulate the 328x printers.
- ▶ There is no mechanism for the Telnet client to request a 3270 device name.
- ▶ ATTN and SYSREQ keys are not always supported.
- ▶ SNA positive/negative response, Sysreq, Start Data Indicator, BID, Signal or Sense Data are not supported.

Note: z/OS Telnet server support the ATTN key both in TN3270 and TN3270E.

In order to solve these issues, TN3270E was introduced. Telnet clients and servers negotiate the support of TN3270E. If either side does not support TN3270E, traditional TN3270 can be used.

Once both sides have agreed on using TN3270E, they begin to negotiate the subset of TN3270E options. These options are device-type and a set of supported 3270 functions:

- ▶ 328x printer support
- ▶ Device name specification

A TN3270E client can optionally request that a particular device name be assigned. If the requested device name is allowed for this client (based on LU mapping statements) the session is established. Otherwise, the session is rejected.
- ▶ The passing of BIND information from server to client
- ▶ Positive/negative response exchange

Support of SNA responses all the way to the client helps synchronize data flow and provides a more accurate measurement of user response. The following response types are supported:

 - Definite
 - Exception
 - No response
- ▶ Sysreq function
- ▶ Contention resolution

Improves communication between the client and host VTAM applications. It includes the following:

 - Start Data Indicator (SDI)
 - BID
 - Signal Indicator

► SNA Sense Support

RFC 1647 has already been made obsolete by RFC 2355. The updates are mostly the clarification of RFC 1647, but it added new DATA-TYPE - PRINT-EOJ. z/OS Telnet provides the PRINT-EOJ function to synchronize SNA brackets with the Telnet printer client.

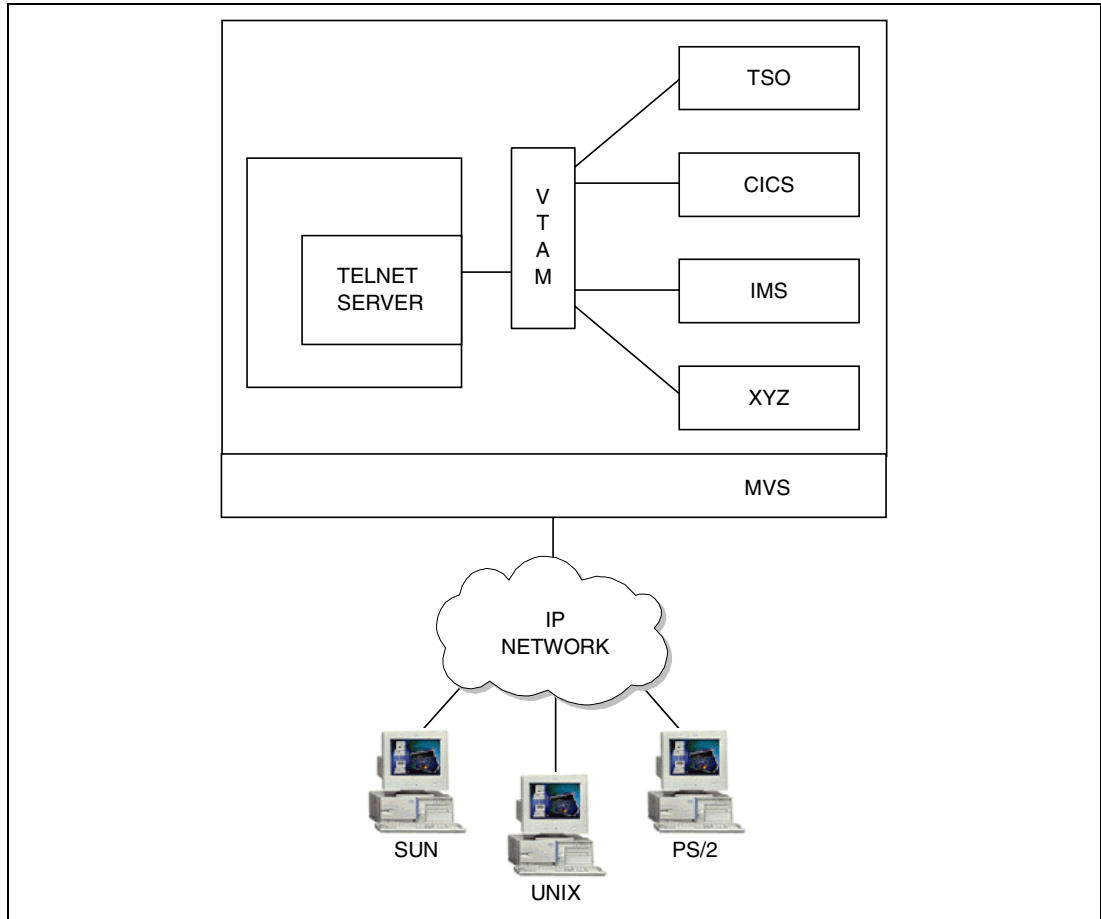


Figure 8-1 Telnet overview

8.1.1 Telnet functions

z/OS Telnet provides:

- Client and server support for TN3270 full-screen (transparent) mode
- Server support for TN3270E mode as in RFC 1647
- Client and server support for line mode
- Server support for 3270 DBCS transform mode, which supports full-screen access from a VT100 or VT282 remote Telnet client

The Telnet server has been completely rewritten to take advantage of MVS service request block (SRB) and multiple task control block (TCB) capabilities on OS/390 V2R5 IP and later. The following sections explain some of the features you find in the Telnet component of CS for z/OS.

Secure sockets support

Secure sockets support for the z/OS Telnet server provides secure data transmission between a secure sockets port and a Secure Sockets Layer (SSL) enabled Telnet client.

In an SSL-encrypted session, any data on a secure port is encrypted using the SSL protocol before it is sent to the client. Data received from the client is decrypted before the data is sent to other processes, such as VTAM. The flows between Telnet and VTAM are unchanged.

z/OS CS IP also supports SSL client authentication which allows additional authentication and access control checking. CS for OS/390 V2R8 IP used the key ring utility GSKKYPAN for managing keys. Since CS for OS/390 V2R10 IP, SSL is supported by the system security environment (z/OS Cryptographic Services), which uses RACF panels and commands. Additionally, CS for OS/390 V2R10 IP included some enhancements regarding connection type (SSL and negotiated security are included).

A key ring file is used to obtain certificate/key information. The file may be an MVS data set (using RACF) or a Hierarchical File System (HFS) file (using GSKKYPAN).

If you need some more information about the concepts of cryptography and SSL, you can find some additional information in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 and at the following Web sites:

- ▶ About SSL protocol:
<http://home.netscape.com/eng/ssl3/ssl-toc.html>
- ▶ About the encryption methodology:
<http://www.verisign.com/repository/crptintr.html>
<http://www.verisign.com/client/about/introCryp.html>

Multiple ports support

CS for z/OS IP provides the capability for the Telnet server to listen on multiple ports. This means you can define different security levels (basic, secure, negtsecure, all) and/or different configuration parameters for each port. Up to 255 ports are allowed. You define ports using statements within the TELNETPARMS and BEGINVTAM information blocks. You can use the **VARY** and/or **DISPLAY** operator commands to select a particular type of port access or a specific port to display related information.

Parameter group mapping

This mapping function gives the ability to map certain parameters to clients based on client identifier. This enables the configuration of different parameter groups containing different security specifications for different clients.

TN3270E mode

The Telnet server supports this connection mode, which simulates most closely a real SNA connection, including LU0/LU2 support for display session and LU1/LU3 support for printer session.

The z/OS Telnet supports RFC 1647. TN3270E offers support for device name specification, 3287-type printer emulation and additional SNA function and protocols. The range of support for RFC 1647 in the clients varies and is negotiated at connection time. The client must also support RFC 1647 in order for TN3270E to be used.

Unformatted System Services (USS) support

USS support provides the ability to emulate the VTAM USS messages. You may define one or more USS tables. The actual USS table to be used for the Telnet sessions can be selected via the USSTCP definition. USS table support has been enhanced with OS/390 V2R5 IP and later and now supports the **LOGON APPLID() DATA() LOGMODE()** command. NQN is available on USSCMD in CS for OS/390 V2R8 IP. NQN application names can be used as DEFFAULTAPPL and LINEMODEAPPL.

Client identifier (CLID) to object mapping

The CLID to object mapping function provides the capability to select both an LU name and an application name for incoming Telnet sessions. The selection may be made on the basis of a specific IP address, a group of IP addresses, a subnet, or the link name used to connect to the z/OS. The function makes the LU name and the application name predictable and controllable. In addition, OS/390 V2R7 IP supports LU and application name selection with a host name or a group of host names. Some enhancements in CS for OS/390 V2R10 IP were to support multiple LU or LU group mappings to the same IP address or IP address group (client identifier).

Takeover functionality

The Telnet takeover function enables the reconnection of a client and an SNA session after the recovery of a failed TCP connection between the client and Telnet server. In this case, the client must specify the specific LU name of the session.

Support for 3270 DBCS transform

The 3270 DBCS transform mode provides 3270 full-screen emulation, but data exchange between the client and the z/OS Telnet server is in ASCII. So all the needed translation takes place in z/OS. The client may emulate VT100 or VT282 type terminal.

SMF reporting

New record type 119 was introduced by CS for z/OS V1R2 IP and is controlled by use of the TYPE119/NOTYPE119 operands on the SMFINIT and SMFTERM statements. Record type 118 is also supported. SMF records may be recorded for the Telnet server and client. The consistency and completeness of record contents allows service providers to better monitor service level agreements and provide usage billing to their customers.

DEBUG utility

OS/390 V2R10 introduced a DEBUG function to aid in tracking state changes and to provide connection ID and LU name information.

AbendTrap command

New in z/OS V1R2 CS IP, the AbendTrap can be used to set up and abend based on some variables specified in the command. The command is:

```
VARY TCPIP, ,TELNET,ABENDTRAP,module,rcode,instance
```

Console commands

CS for z/OS IP supports several console commands. The operator may issue **DISPLAY** or **VARY** commands to monitor and control z/OS Telnet. The new commands offer more consistency with other products such as VTAM, provide more information, and take advantage of MCS.

Binary option for line mode

Some applications running in line mode require the ability to do their own unique translation of the data in the line mode packet. Normally the Telnet server will automatically translate the data from EBCDIC to/from ASCII. Clients working in line mode may bypass this translation.

Generic resources

Generic resources allows for the concurrent multiplicity of VTAM applications in the z/OS environment. When requests come in for the application, VTAM chooses the optimal application instance for the client.

Workload Manager (DNS/WLM) support

Several applications, including the Telnet server shipped with CS for z/OS, are able to register with WLM for load distribution. Relying on the CS for z/OS IP DNS, names are created dynamically when Telnet registers to WLM. The host name to IP address mapping enables the DNS server to distribute load during name resolution of each client.

Sysplex Distributor support

CS for z/OS IP also provides the Sysplex Distributor function, which provides for connection dispatching of a particular service. Sysplex Distributor can be used with the Telnet server for load distribution among many Telnet servers in a sysplex. Unlike DNS/WLM, Sysplex Distributor dispatches connections at the IP layer, thereby overcoming the many challenges in DNS-based load distribution techniques.

Note: Load distributions with DNS/WLM and Sysplex Distributor work nicely if there is no GR instance in the same z/OS where the Telnet server resides. If there is, the local VTAM will always choose a zero-hop instance. This is because Telnet terminals are applications for VTAM. OS/390 V2R5 and above provides new options to bypass VTAM's bias to choose the nearest instance when resolving a GR name. It does so by allowing you to change the coding of the GR resolution exit, ISTEYCGR.

Automatic VIPA Takeover/Takeback

The Automatic VIPA Takeover/Takeback function allows the Telnet server to provide for higher availability in a sysplex environment. A dynamic VIPA can be coded to represent the Telnet service. This VIPA can move from one stack to another within a sysplex in the event the Telnet server fails, thus providing increased availability of the Telnet service.

Profile replacement

When you replace a profile with the **VARY OBEYFILE** command, new PROFILE statements completely replace the PROFILE statements in effect before an update on a port basis. The updates are not cumulative from the previous PROFILE.

The most recent profile is referred to as the current profile. New connections use the tables generated by the current profile, but the tables generated by older profiles remain active and continue to support any connections established when the older profiles were the current profile.

8.1.2 Telnet printer support

Telnet clients can route SNA print output over TCP/IP. 328x class printers (device type IBM-3287-1) are supported by z/OS Telnet. These printers support both the SNA character stream (LU Type 1) and 3270 data stream (LU Type 3). The Telnet printer communicates with VTAM applications as a secondary LU and looks exactly like an actual 3287-class printer. The server will drop the connection if a printer request comes in during negotiation as a non-TN3270E connection.

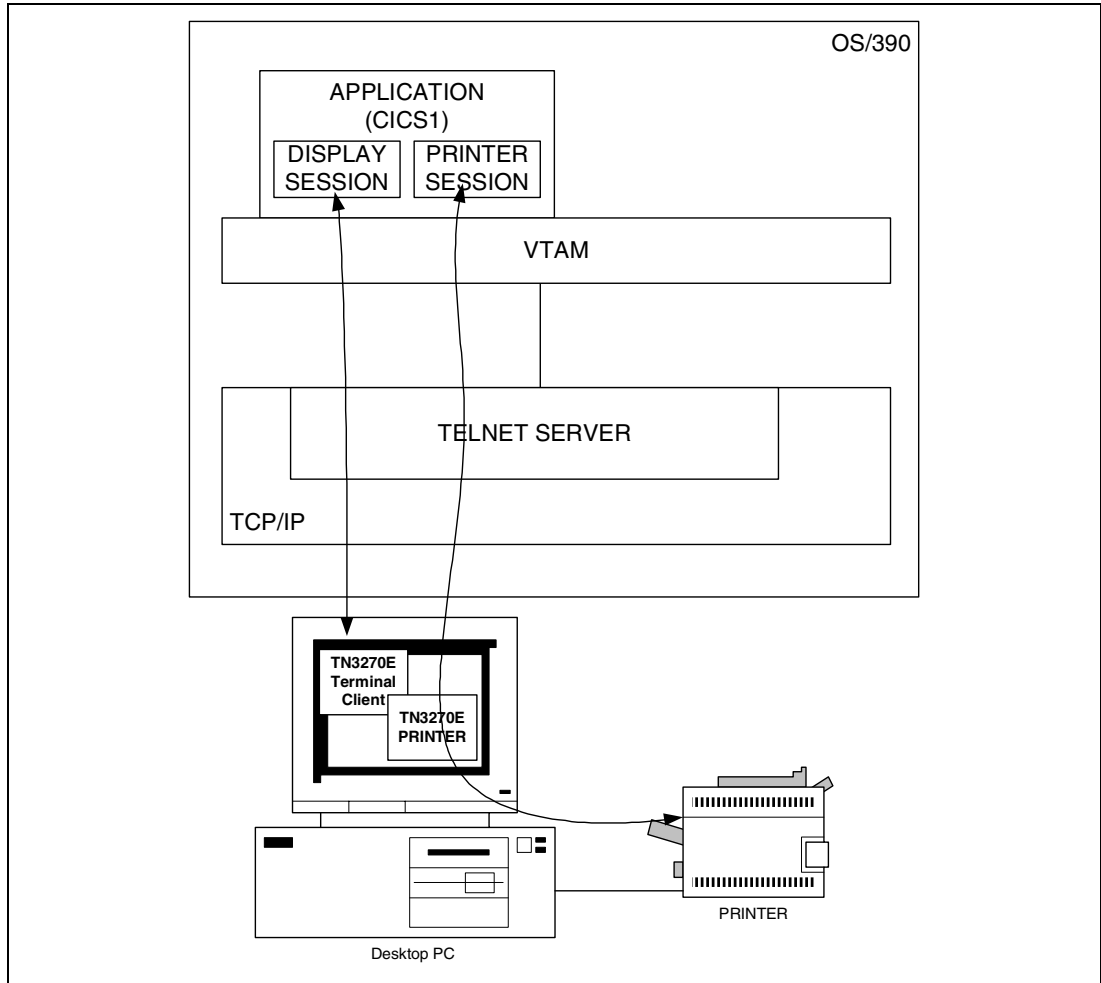


Figure 8-2 z/OS Telnet with TN3270E printer emulation

Telnet 3287 printer emulation allows the Telnet administrator to use a single product (that is, Telnet) to route SNA print output over TCP/IP. Figure 8-2 shows the simplicity of this solution.

Printer association

Once printer emulation is available, it is useful to associate printer device names with terminal device names. With association, end users can connect their Telnet terminals to an application and then associate their Telnet printer device with the terminal device. For example, a CICS table may specify that if a terminal LU is requesting a printer function, the output should be routed to the printer that is associated with that terminal LU.

Note: To utilize this function, the Telnet server needs to work together with the Telnet client. Therefore, the client also has to support printer association.

FM Header support

z/OS Telnet supports FM Header (FMH). Since RFC 1647 does not specify exact requirements about FMH, some of the initial implementations of TN3270E had a problem with FMH support. FMH is used with the LU1 printer stream.

PRINT-EOJ support

PRINT-EOJ is not a part of RFC 1647 but a part of RFC 2355, which made RFC 1647 obsolete. In native SNA, bracket control is used to indicate the beginning or end of a group of exchanged requests. But since the Telnet session does not pass request and response header (RH) information, there is no way for the Telnet client to notice the end bracket. This means that the Telnet printer has to hold the actual printing until the obvious end of printing, that is UNBIND. With some kinds of applications, this implementation may result in client hangs.

To solve this problem, PRINT-EOJ will be sent to the client when the server receives an end bracket from the application.

8.2 Telnet server customization

Some of the options you have when configuring Telnet are covered in this section:

- ▶ 8.2.1, “Customizing the TCP/IP procedure”
- ▶ 8.2.2, “Customizing the VTAM configuration data set”
- ▶ 8.2.3, “Customizing the PROFILE data set”
- ▶ 8.2.4, “CLID to object mapping”
- ▶ 8.2.5, “USS messages”
- ▶ 8.2.6, “Using translation tables”

Follow the steps to configure the Telnet server as explained in *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776.

8.2.1 Customizing the TCP/IP procedure

Telnet Server is part of TCP/IP stack in z/OS CS IP. When you start TCP/IP, Telnet Server is available if there is a PROFILE statement for that (see 8.2.3, “Customizing the PROFILE data set” on page 185).

If you want to provide full-screen access to Telnet from non-3270 terminals such as VT100 and VT282, you need to configure the Telnet server for 3270 single-byte character set (SBCS) or double byte character set (DBCS) transform mode. You need to include special data definition (DD) statements in the TCPIP cataloged procedure and specify the DBCSTRANSFORM parameter in the TELNETPARMS statement in the TCPIP PROFILE data set. The following are the sample DD definitions for DBCS transform mode:

```
//TNDBCSCN DD DSN=h1q.SEZAINST(TNDBCSCN),DISP=SHR ❶  
//TNDBCXSL DD DSN=h1q.SEZAXLD2,DISP=SHR ❷  
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136) ❸  
:  
//SYSTCPD DD DSN=TCP.TCPPARMS(TDATA&SYSCLONE.A),DISP=SHR ❹
```

Figure 8-3 DBCS related DD statements in the TCP/IP procedure

In this example:

❶ The TNDBCSCN DD must point to the configuration data set for the 3270 DBCS transform mode. This configuration data set specifies the default DBCS conversion mode that will take effect at initialization time.

Specify the CODEKIND and CHARMODE parameters according to the required DBCS code page. A sample configuration can be found in the installation data set h1q.SEZAINST(TNDBCSCN).

2 The TNDBCSXL DD must point to the data set containing binary translation table code files for the 3270 DBCS transform mode. The installation data set hlq.SEZAXLD2 contains the default binary translation table code files.

3 When DBCSTRACE is specified in the PROFILE data set, the debug output from the 3270 DBCS transform mode is sent to the location specified in the SYSPRINT output DD statement.

4 The SYSTCPD DD is still needed for the Telnet server. The environment variable "RESOLVER_CONFIG" will not be used by Telnet.

8.2.2 Customizing the VTAM configuration data set

The Telnet server needs VTAM definitions to enable Telnet clients to access SNA applications. Since the Telnet server itself is a VTAM application, you have to define an application program major node with minor nodes that represent Telnet clients.

To ensure that the VTAM definition is enabled when VTAM is started, you may want to update the VTAM configuration data set. You can do this by specifying the name of the VTAM application program major node to the ATCONNxx member of your VTAMLST.

To configure the application program major node, copy the sample provided in hlq.SEZAINST(VTAMLST) and modify it to suit your installation.

When you update the VTAM configuration data set, you can define Telnet LUs to VTAM using a wild card character. VTAM Version 4 Release 3 introduced a new function called model application names. This function gives system administrators the ability to code a generic APPL name using the * or ? character. With this capability, Telnet administrators can use a simpler definition with the TCP/IP range statement coding (for example, LU00001..LU20000).

Both TCP/IP and VTAM support system symbolics. Figure 8-4 shows an example.

```
*
* VTAMLST SAMPLE DEFINITION
*
TELAPPL  VBUILD TYPE=APPL
RA&SYSCLOE.TN?? APPL AUTH=NVPACE,EAS=1,PARSESS=NO,           X
                MODETAB=ISTINCLM,SESSLIM=YES
RA&SYSCLOE.TP?? APPL AUTH=NVPACE,EAS=1,PARSESS=NO,           X
                MODETAB=ISTINCLM,SESSLIM=YES
```

Figure 8-4 VTAM definitions for Telnet with system symbolics

Note: Because the TCP/IP LU code cannot handle multiple concurrent sessions, make sure that the SESSLIM=YES parameter is coded for each VTAM APPL statement to ensure correct Telnet processing with applications using VTAM CLSDST/PASS macros. Also, code EAS=1 to minimize CSA storage use and LOSTERM=IMMED to ensure the quickest Telnet LU ACB cleanup.

The default LOGMODE entries are shown in the table of Telnet device name parameters in *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776. The logmodes used can be changed with the TELNETDEVICE statement for both TN3270 and TN3270E sessions. Be sure to code both the TN3270 and TN3270E logmodes if the same logmode is desired for connection types. It is common to leave off the second, TN3270E, logmode specification with the result being that all TN3270E connections use the default logmode.

For further SNA details the following books are recommended:

- ▶ *z/OS V1R2.0 Communications Server SNA Network Implementation*, SC31-8777
- ▶ *z/OS V1R2.0 Communications Server SNA Resource Definition Reference*, SC31-8778

8.2.3 Customizing the PROFILE data set

The TCP/IP PROFILE data set is used to configure the Telnet server. You can update the PROFILE data set to change or add statements. You can change the association of VTAM LUs and IP addresses or host names, add host printers, or switch the port(s) used by Telnet. The Telnet server uses this information to build its own internal profile. This Telnet profile can be replaced easily without stopping and restarting the TCP/IP stack using the **VARY OBEYFILE** command. Profiles are managed in the following manner:

Complete replacement

New profile statements completely replace old profile statements. The updates are not cumulative from the previous profile. Old profiles are referred to by sequential numbers starting with 1. The new profile is referred to as the current profile.

In general, Telnet uses the last valid value or statement that was specified. However, if the replacement statement does not contain all of the required parameters or the required parameter contains errors in all of its list elements, the statement being replaced will be removed and the replacement will not occur.

The only exception to the rule is the IP addresses definition in the IPGROUP statement. In this example:

```
IPGROUP ABC 1.1.1.1 2.2.2.2
IPGROUP XYZ 2.2.2.2 3.3.3.3
```

The second IPGROUP statement will receive an error message indicating:

```
2.2.2.2 already defined in an IPGROUP, it is ignored.
```

Connection association

New connections use the current profile. Older profiles remain active and continue to support old connections.

Profile table management

When all connections associated with an older profile have ended, the storage for the profile tables is freed, and the profile is now inactive. This is not a problem because all new connections must use the current profile.

Notes:

1. Transform is supported only on a single port. To change the port to use transform, the port must be terminated first. Then a **VARY OBEY** can be used to define transform support to another port.
2. When the profile is read by Telnet, the Workload Manager names are registered. Registration names must be duplicated in later Telnet profiles to keep the registration. Otherwise, the name will be deregistered from DNS. This only applies to DSN/WLM.
3. When you do a **VARY OBEYFILE** to update the Telnet profile, the profile must include both a TELNETPARMS block and a BEGINVTAM block for each port. If either of the blocks is not present, the profile update does not occur for the port.

Stand-alone PORT and PORTRANGE statements

Prior to OS/390 V2R5 IP, the Telnet server was configured with a stand-alone port reservation statement such as PORT 23 INTCLIEN and the BEGINVTAM/ENDVTAM group of statements. This is still supported and if a single port is used, the first stand-alone INTCLIEN port statement will generate a TELNETPARMS block with all default values.

It is *not* necessary to code PORT or PORTRANGE statement to start the Telnet server. But you may reserve Telnet ports if there is any possibility of another application taking the ports that Telnet will use. Figure 8-5 shows an example of Telnet port reservation:

```
PORT
  23    TCP INTCLIEN ; Telnet Server basic port
  :
PORTRANGE
  623 7 TCP INTCLIEN ; Telnet Server secure ports
```

Figure 8-5 Telnet port reservation with INTCLIEN

If you are using multiple Telnet ports, each port must be defined to Telnet with a TELNETPARMS block. Even if using one port, a TELNETPARMS block is recommended.

Note: When a TELNETPARMS PORT statement is present with the stand-alone PORT statement, the TELNETPARMS information will be used. The port numbers should match. If they do not match, the TELNETPARMS PORT number will be used and the stand-alone PORT number is reserved for no use.

TELNETGLOBALS statement

TELNETGLOBALS statement is optional and contains Telnet parameters statements that define port characteristics across all ports. Table 8-1 on page 187 shows a list of Telnet parameters and where they can be coded.

You can see an example configuration with TELNETGLOBALS in Figure 8-6. You can find more information in *z/OS V1R2.0 Communications Server IP Configuration Guide*, SC31-8775.

```
TELNETGLOBALS
  KEYRING hfs /usr/keyring/tcps.kdb ;keyring used by all SECUREPORTs
ENDTELNETGLOBALS

TELNETPARMS      ; this secure port definition uses TELNETGLOBALS
  SECUREPORT 992 ; KEYRING definition
ENDTELNETPARMS
```

Figure 8-6 TELNETGLOBALS Telnet server profile definition

TELNETPARMS statement

This is a required statement where the characteristics of a specified port are coded. Figure 8-7 shows an example.

```

TELNETPARMS
  PORT 23
  INACTIVE 600
  TIMEMARK 600
  SCANINTERVAL 120
  SMFINIT STD
  SMFTERM STD
  WLMCLUSTERNAME TN3270E ENDWLMCLUSTERNAME
ENDTELNETPARMS

```

Figure 8-7 TELNETPARMS statement example


Notes:

1. If no statements are entered between TELNETPARMS and ENDTELNETPARMS, Telnet will use the default values.
2. If there are duplicated statements within the TELNETPARMS block, the last statement with no errors is used.

You can define as many as 255 listening ports for session requests. Ports are defined using PORT or SECUREPORT statements within the TELNETPARMS information blocks. Each port requires a separate TELNETPARMS block.

Table 8-1 shows a list of Telnet parameters and where they can be coded. An X in a column indicates that the parameter can be coded in the indicated block. For example, CLIENTAUTH can be coded in TELNETGLOBALS, TELNETPARMS or PARMSGROUP (affecting all connections on all ports, all connections on one port or a subset of connections on one port, respectively). For a description of each parameter, see *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776.

Table 8-1 Telnet parameter statements

Statement	TELNET GLOBALS	TELNET PARMS	PARMS GROUP	BEGINVTAM
BINARYLINEMODE		X		
CLIENTAUTH	X	X	X	
CODEPAGE	X	X	X	
CONNTYPE	X	X	X	
CRLLDAPSERVER	X			
DBCSTRACE		X		
DBCSTRANSFORM		X		
DEBUG	X	X	X	
DISABLESGA		X		
ENCRYPTION	X	X	X	
EXPRESSLOGON NOEXPRESSLOGON 	X	X	X	
FULLDATATRACE NOFULLDATATRACE	X	X	X	
INACTIVE	X	X	X	

Statement	TELNET GLOBALS	TELNET PARMS	PARMS GROUP	BEGINVTAM
KEEPINACTIVE	X	X	X	
KEYRING	X	X		
LUSESSIONPEND NOLUSESSIONPEND	X	X	X	X
MAXRECEIVE		X		
MAXREQSESS		X		
MAXVTAMSENDQ		X		
MSG07 NOMSG07	X	X	X	X
OLDSOLICITOR		X		
PORT/SECUREPORT		X		
PRTINACTIVE	X	X	X	
SCANINTERVAL TIMEMARK	X	X	X	
SECUREPORT		X		
SIMCLIENTLU NOSIMCLIENTLU	X	X	X	
SINGLEATTN		X		
SMFINIT SMFTERM	X	X	X	
SNAEXT NOSNAEXT ¶	X	X	X	
SSLTIMEOUT	X	X	X	
TESTMODE		X		
TIMEMARK	X	X	X	
TKOSPECLU		X		
TKOSPECLURECON		X		
TN3270E NOTN3270E	X	X	X	
WLMCLUSTERNAME		X		

¶ - new parameters in z/OS V1R2 CS IP.

BEGINVTAM and ENDVTAM statements

When you define general characteristics for the Telnet port using the TELNETPARMS statements, you have completed half of the necessary profile configuration tasks. The other half is completed when you define LUs, printers, applications, and mapping statements.

To do this, you add statements to the BEGINVTAM information block in the PROFILE data set. The BEGINVTAM statement indicates the start of the list of valid VTAM statements to configure the Telnet server. The ENDVTAM statement ends the list of VTAM parameters. Figure 8-8 shows an example.

```

TELNETPARMS 1
  PORT 23
ENDTELNETPARMS

TELNETPARMS 1
  SECUREPORT 992 KEYRING MVS TCPCS.KEYRING
ENDTELNETPARMS

BEGINVTAM
  PORT 23 992 2

  TelnetDEVICE 3278-2 D4B32782,SNX32702 ; 24 x 80
  TelnetDEVICE 3278-2-e NSX32702,SNX32702 ; 24 x 80
  TelnetDEVICE 3278-3 D4B32783,SNX32703 ; 32 x 80
  TelnetDEVICE 3278-3-e NSX32703,SNX32703 ; 32 x 80

  LUGROUP sysusr1u CONSLU1..CONSLU7 ENDLUGROUP
  LUGROUP
    acct1u ACCLU01 ACCLU03 ACCLU05 ACCLU07 ACCLU08
    ACCLU16 ACCLU27 ACCLU38 ACCLU49 ACCLU68
    ACCLU71 ACCLU75 ACCLU78 ACCLU84 ACCLU95
  ENDLUGROUP

  IPGROUP
    admin 255.255.240.0:198.25.32.0
  ENDIPGROUP

  LUMAP acct1u admin
  LUMAP sysusr1u 198.25.32.10
  LUMAP CONSLU4 199.28.33.12

  DEFAULTAPPL ADMLOGO admin
  DEFAULTAPPL CADLOGO 198.25.32.10
  DEFAULTAPPL ANONLOGO ETH1

  LINEMODEAPPL APCLOGO admin
  LINEMODEAPPL ANZLOGO 198.25.32.10
  LINEMODEAPPL NONLOGO ETH1

  RESTRICTAPPL CADLOGO
  USER NIGEL
  LU CONSLU1 LU CONSLU2 LU CONSLU3
  ALLOWAPPL ANZLOGO
  LU CONSLU4 LU CONSLU5 LU CONSLU6 LU CONSLU7

  USSTCP ABCLOGON TR1
  USSTCP CBALOGON 198.25.32.11
  USSTCP CBALOGON 199.28.33.12
ENDVTAM

```

Figure 8-8 BEGINVTAM and ENDVTAM statements

In the example:

1 Multiple TELNETPARMS blocks are supported resulting in a basic and a secure port.

2 The PORT statement in the BEGINVTAM block allows multiple or range-based port definitions. You can define a single BEGINVTAM block for multiple TELNETPARMS blocks. Multiple BEGINVTAM blocks are also supported.

Table 8-2 shows BEGINVTAM statements supported in CS for z/OS. Some statements can be used only with higher OS/390 releases. These are shown with the OS/390 release number.

Table 8-2 BEGINVTAM statements in CS for z/OS IP

ALLOWAPPL	The ALLOWAPPL statement is used to specify which VTAM application names clients can access. You can use the full application name or application names with wild cards for mapping application to LU or LUGROUP (OS/390 V2R10 or higher). For example ALLOWAPPL * allows all applications.
DEFAULTPRTSPEC V2R10	The DEFAULTPRTSPEC statement defines a default list or range of LUs. This pool can be used by a specific printer connection request if no other LU mapping method is attempted for the client.
DEFAULTPRT V2R10	The DEFAULTPRT statement defines a default list or range of LUs. This pool can be used by a generic printer connection request if no other LU mapping method is attempted for the client.
DEFAULTAPPL V2R7	The DEFAULTAPPL statement specifies the initial application to which to connect when a Telnet client establishes a connection other than line mode. OS/390 V2R10 and higher provide more enhancements regarding DEFFAULTAPPL which are FIRSONLY, DEFONLY and LOGAPPL parameters.
DEFAULTLUS	The DEFAULTLUS statement defines a list or range of default LUs. This pool can be used by a general terminal connection request if no other LU mapping method is attempted for the client.
DEFAULTLUSSPEC V2R10	The DEFAULTLUSSPEC statement defines a list or range of default LUs. This pool can be used by a specific terminal connection request if no other LU mapping method is attempted for the client.
DESTIPGROUP z/OS V1R2	DESTIPGROUP client identifier statement is used to define a group of destination IP addresses. The group name can be used on several mapping statements.
HNGROUP V2R7	The HNGROUP statement defines a group of host names. Wild card names are accepted.
IPGROUP	IPGROUP client identifier statement defines a group of IP addresses. Each group name can be used on several mapping statements.
INTERPTCP V2R7	The INTERPTCP statement maps a customized interpret table to an IP address, a host name or a network interface. This table is used to interpret incoming USS commands before the USS command processor is invoked. If the input string does not match any interpret table entry, the USS command processor parses the input string.
LINEMODEAPPL V2R7	The LINEMODEAPPL statement provides default application to line mode client connections.
LINKGROUP z/OS V1R2	LINKGROUP client identifier statement defines a group of link names. The group name can be used on several mapping statements.

LUGROUP	LUGROUP statement defines a group of LUs. These group names can be used on the LUMAP statement to represent an LU pool.
LUMAP V2R7	<p>The LUMAP statement maps LUNAMES or LUGROUPS to clients that are specified by client identifiers parameter. For a list of client identifiers see Table 8-3 on page 193.</p> <p>OS/390 V2R10 and higher support more LUMAP functions including multiple LUMAP mapping to same client. Parameters included by OS/390 V2R10 were LOGAPPL, KEEPOPEN and FIRSTONLY.</p> <p>LOGAPPL application_name optional keyword specifies the initial application to which Telnet will connect. In addition, if the application is not available, Telnet will respond with MSG07 to the client and keep the LU ACB open and ready to accept a BIND from the application.</p> <p>Specifying the KEEPOPEN keyword means that all LUs identified in the lu_group_name or the LU identified by lu_name will always have an OPEN ACB as long as the connection exists.</p> <p>What happens at session logoff depends on whether FIRSTONLY is coded. If FIRSTONLY is not coded, Telnet will issue another Request Session to the default application (defined by either LUMAP-DEFAPPL or DEFAULTAPPL). If FIRSTONLY is coded, Telnet will send a USSMSG10 screen or Solicitor Panel to the client.</p>
LUSESSIONPEND	LUSESSIONPEND parameter statement allows the server to redrive the DEFAULTAPPL, USS, or Solicitor screen after LOGOFF of the current session. If this statement is not coded or NOLUSESSIONPEND is specified, the Telnet server connection is dropped after session LOGOFF.
MSG07 V2R7	MSG07 statement is used to activate logon error message processing. Specifying this statement will provide information to the client when a session attempt to the target application fails.
PARMSGROUP V2R10	The PARMSGROUP statement defines parameters that will be mapping to certain clients. The PARMSGROUP parameters are mapped to client identifiers with the PARMSMap statement. For connections that map to a PARMSGROUP, the statements defined in the PARMSGROUP block override those defined in the TELNETGLOBALS, TELNETPARMS, or BEGINVTAM block.
PARMSMAP (OS/390 V2R10)	The PARMSMAP statement maps PARMSGROUP parameters to specified clients which are described with a client identifier. For a list of client identifiers, see Table 8-3 on page 193.
PORT V2R6	The PORT statement must be used to associate the BEGINVTAM block with the correct TELNETPARMS block when multiple ports are used. The PORT statement must be the first statement in the BEGINVTAM block. You can define separate BEGINVTAM blocks for each TELNETPARMS port or you can use only one BEGINVTAM block with all port numbers for all TELNETPARMS ports.
PRTGROUP	The PRTGROUP statement defines a printer LU group for TN3270E printer support.
PRTMAP	The PRTMAP statement maps a printer or a printer group to a client identifier. For a list of client identifiers, see Table 8-3 on page 193.

QUEUESESSION	QUEUESESSION parameter statement signifies that all DEFAULTAPPL applications queue their sessions within VTAM when performing a CLSDST-PASS. At session logoff, Telnet leaves the LU ACB open and waits for a BIND from the DEFAULTAPPL application.
RESTRICTAPPL	RESTRICTAPPL statement restricts access to the specified application. This statement must be followed by user parameters defining each user who is authorized to use the application. Users are prompted to identify themselves with a password. RACF or an equivalent security program is used to validate the password. If no user parameters are specified, the application cannot be accessed.
TELNETDEVICE	This statement defines logmode entries for TN3270 and TN3270E devices. You should use a TELNETDEVICE statement for each type of terminal and one TELNETDEVICE statement for printers. Logmode entries define device type, enhanced facility support capability (TN3270 or TN3270E), terminal screen size, alternate terminal screen size capability and size, etc.
USERGROUP z/OS V1R2	The USERGROUP object statement defines a group of user IDs. The group name can be used on several mapping statements.
USSTCP	The USSTCP statement allows you to map a customized USS table to either a remote host name, remote IP address, or a network interface. USSTables of USSTCP can be any non-SNA USSTables.

For a complete description of all parameters see *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776.

The BEGINVTAM ENDVTAM block contains different characteristics in the same block, for example display and printer LU definitions, application characteristics, address mapping definitions, and so on.

The following are the major categories in a block:

- ▶ Device-type to logmode mapping
 - TELNETDEVICE
- ▶ Initial screen selection
 - DEFAULTAPPL
 - LOGAPPL (With LUMAP or DEFAULTAPPL in OS/390 V2R10)
 - LINEMODEAPPL
 - USSTCP
 - INTERPTCP
- ▶ Application permission
 - ALLOWAPPL
 - RESTRICTAPPL
- ▶ LU to IP address or host name mapping
 - DEFAULTLUS
 - DEFAULTPRT(OS/390 V2R10)
 - DESTIPGROUP (z/OS V1R2)
 - LINKGROUP (z/OS V1R2)
 - LUGROUP

- PRTGROUP
- IPGROUP
- HNGROUP
- LUMAP
- PRTMAP
- USERGROUP (z/OS V1R2)
- ▶ Connection handling
- PORT
- MSG07
- LUSESSIONPEND
- QUEUESESSION

There is also the INCLUDE statement that can be used to insert and process configuration statements in the PROFILE.TCPIP data set. This statement causes profile statements from the named data set to be included at the point that the INCLUDE statement is encountered.

8.2.4 CLID to object mapping

Telnet mapping can be used to improve control of Telnet access relative to application security and routing through the correlation of client identifiers and objects.

Application security means that you are able to control the mapping of Telnet client IP addresses or host names to specific LU names. You can define groups of IP addresses, host names, and LU names. You can map an LU name or an LU name group with some client identifiers, as described on Table 8-3.

Table 8-3 Client identifier types and definitions

Client Identifier Type	Definition
UserID	The Client User ID derived from the client certificate at connection time when ClientAuth SAFcert is specified on an SSL connection.
Hostname	The completely qualified client host name.
IPAddr	The client IP address expressed in dotted decimal form.
UserGrp	The USERGROUP name that contains exact or wild-carded client user IDs.
HNGrp	The HNGROUP name that contains exact or wild-carded client host names.
IPGrp	The IPGROUP name that contains exact or subnetted client IP addresses.
DestIPGrp	The DESTIPGROUP name that contains exact or subnetted destination IP addresses.
Linkname	The link name defined by the LINK statement in PROFILE.TCPIP.
DestIP	The destination IP address expressed in dotted decimal form.
LinkGrp	The LINKGROUP object name that contains exact or wild-carded link names.

Client Identifier Type	Definition
NULL	Not coded, but listed here for completeness. The client identifier type indicates that no client identifier was specified. This is valid for the DEFAULTAPPL, LINEMODEAPPL, USSTCP and INTERPTCP mapping statements. It is the implied client identifier for the DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTPRT and DEFAULTPRTSPEC object statements.

When client identifiers are used together, conflicts may occur. To avoid them, Telnet server performs the client identifier selection rules in the following order:

1. Exact client identifier (user ID, host name, IP address):

```
LUMAP LU1 USERID,USER1
LUMAP LU2 NAME1.HOST1.COM
LUMAP LU3 1.2.3.4
```

Client identifier type USERID is required. If not specified, USER1 is assumed to be a link name.

2. Exact client identifier in a group definition (user group, host name group, IP address group):

```
USERGROUP USRGRP1
  USER1 USER2 USER3
ENDUSERGROUP
HNGROUP HNGRP1
  NAME2.HOST1.COM NAME2.HOST3.COM
ENDHNGROUP
IPGROUP IPGRP1
  1.2.3.5 1.2.3.6
ENDIPGROUP
LUMAP LUGRP1 USRGRP1
LUMAP LUGRP2 HNGRP1
LUMAP LUGRP3 IPGRP1
```

3. Wild card match for client identifier in a group definition (user group, host name group, IP address group):

```
USERGROUP USRGRP2
  USER%% TCPU*
ENDUSERGROUP
HNGROUP HNGRP2
  *.HOST2.COM **.HOST3.COM
ENDHNGROUP
IPGROUP IPGRP2
  255.255.0.0: 2.3.0.0
ENDIPGROUP
LUMAP LUGRP1 USRGRP2
LUMAP LUGRP2 HNGRP2
LUMAP LUGRP3 IPGRP2
```

4. Exact destination (destination IP address, link name):

```
DEFAULTAPPL TSO DESTIP, 1.2.3.4
USSTCP USSTAB1 LINK1
```

client identifier type DESTIP is required. If not specified, destination IP address 1.2.3.4 is assumed to be a client IP address.

5. Exact destination in a group definition (destination IP address group, link name group):

```
DESTIPGROUP DSTIPGRP1
  1.2.3.5 1.2.3.6
```

```

ENDESTIPGROUP
LINKGROUP LINKGRP1
  LINK1 LINK2 LINK3
ENLINKGROUP
LUMAP LUGRP1 DSTIPGRP1
LUMAP LUGRP2 LNKGRP1

```

6. Wild card match for destination in a group definition (destination IP address group, link name group):

```

DESTIPGROUP DSTIPGRP2
  255.255.0.0: 1.4.0.0
ENDESTIPGROUP
LINKGROUP LINKGRP2
  LINK* %LINK
ENLINKGROUP
LUMAP LUGRP1 DSTIPGRP2
LUMAP LUGRP2 LNKGRP2

```

7. Null Client ID (DEFAULTAPPL, LINEMODEAPPL, USSTCP, INTERPTCP, DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTPRT, DEFAULTPRTSPEC):

```

DEFAULTAPPL      TSO
LINEMODEAPPL     CICS
USSTCP           USSTAB1
INTERPTCP        INTTAB1
DEFAULTLUS
  LU01 . . LU99
ENDEFAULTLUS

```

Application routing means that you are able to control what the Telnet client initially is presented with when a new Telnet session is established. When a client connects to the Telnet server, the following sequence will occur:

Step 1. LU name selection for TN3270E clients

This step will occur first only with TN3270E clients. A TN3270E client may optionally request that a particular device name be assigned. If the device name is allowed for this client and is available, the client is assigned the requested device name. Otherwise, the request is rejected with an appropriate reason code. The device name can be either an LU name or an LU group name. A sample IBM Personal Communication Client configuration is shown in Figure 8-9 on page 196.

Because of this implementation, device name selection is required during session negotiation. This means the device name will be chosen before the application is chosen.

LU name selection will occur based on the LU mapping sequence shown in Table 8-4 on page 197. DEFAULTLUS will also be chosen for display sessions if all the mappings fail.

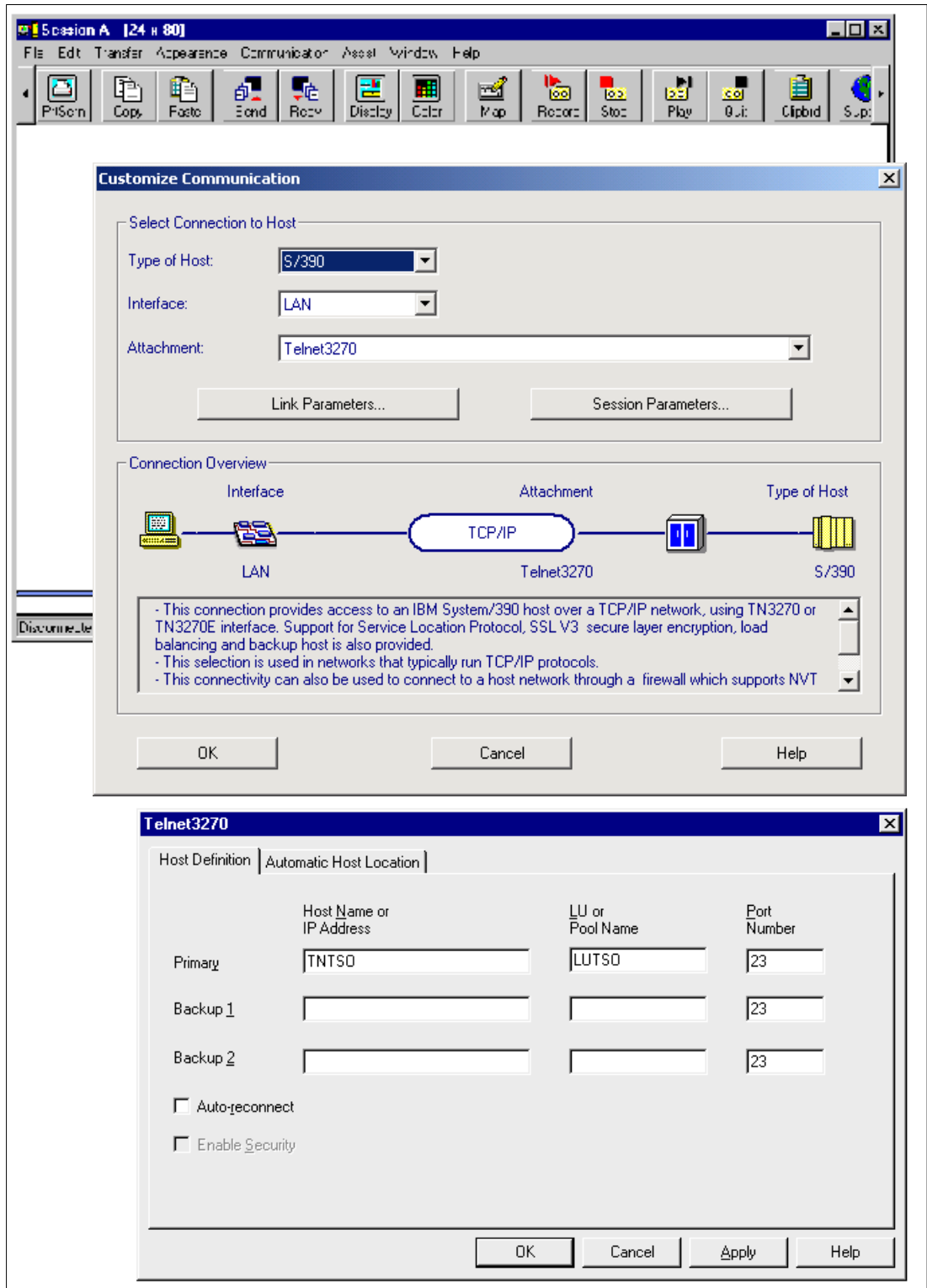


Figure 8-9 Sample client device name configuration

Step 2. Initial screen selection

When a client connects to the Telnet server, it selects an application, a USS message 10 screen, or the Telnet solicitor screen.

Step 3. Application-based LU selection for non-TN3270E clients

After an application is chosen by the user or DEFAULTAPPL definition, LU name selection will occur based on the LU mapping sequence shown in Table 8-4 and LU and LUMAP definition shown in Table 8-5. All the applications must be defined in the PROFILE using the RESTRICTAPPL, ALLOWAPPL, or DEFAULTAPPL statements.

Because of the Step 1 process, TN3270 and TN3270E may result in a different outcome. Consider the example depicted in Figure 8-10.

```
BEGINVTAM
  DEFAULTLUS T1 T2 T3 T4 T5 ENDEFALTLUS
  RESTRICTAPPL APPL1 USER USER54 LU T3
ENDVTAM
```

Figure 8-10 LU selection example

In this example, the server will react differently depending on the client type:

- ▶ TN3270 clients
 - a. Solicitor screen appears.
 - b. Specify APPL1, USER54, and a password. The server selects T3.
- ▶ TN3270E clients
 - a. Solicitor screen appears. The server selects T1 *at this time*.
 - b. Specify APPL1, USER54, and a password. The server fails the connection because of an LU mismatch.

Note: The specification of LU names in the ALLOWAPPL or RESTRICTAPPL is optional. Use of the LUMAP statements is the preferred method of assigning LUs. The RESTRICTAPPL parameter is still viable with a TN3270E connection via the SIMCLIENTLU statement or mapping statements.

Table 8-4 shows the initial screen and LU selection sequence.

Table 8-4 Initial screen and LU selection sequence

Mapped to	Application Security	Application Routing	
	LUMAP	LINEMODEAPPL DEFAULTAPPL	USSTCP ¶
Host Name	1	1	2
IP Address	2	3	4
Host Name Group	3	5	6
IP Address Group	4	7	8
Wildcard Host Name ¶	5	9	10
IP Subnet ¶	6	11	12
Network Interface	not applicable	13	14
No Option	not applicable	15	16

Notes:

- 1 USSTCP is not applicable to the line mode clients.
- 2 Wild cards are not allowed on LUMAP, PRTMAP, DEFAULTAPPL, LINEMODEAPPL, and USSTCP statements. HNGROUP block should be used.
- 3 IP subnets are not allowed on LUMAP, PRTMAP, DEFAULTAPPL, LINEMODEAPPL, and USSTCP statements. IPGROUP block should be used.

Table 8-5 shows the LU name allocation rules based on an application name.

Table 8-5 LU name allocation rules

	RESTRICAPPL/ALLOWAPPL	DEFAULTAPPL
Both LU and LUMAP are defined	The selected LU must be in both lists.	Not applicable
Either LU or LUMAP is defined	The LU is selected from the defined list.	The LU is selected from the LUMAP list.
Neither LU nor LUMAP is defined	The LU is selected from the default list.	The LU is selected from the default list.

LU mapping for application security

You can use LUMAP statements to define a mapping from an LU or a group of LUs to a client identifier. Multiple LUMAP statements for the same client are supported. You can map different LU groups to the same client or client group with IP address or host name. For LUMAP syntax, see *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776.

The LUMAP syntax has expanded throughout the releases to handle more mappings of LUs to CLIDs. Here are some examples of LUMAP coding:

1. Map an LU name with an IP address

In the following example, workstation 9.24.105.220 will always be assigned to an LU name of RA03TN70:

```
LUMAP RA03TN70 9.24.105.220
```

2. Map a group of LUs with an IP address

In order to associate a group of LUs and an IP address, you have to use LUGROUP/ENDLUGROUP statements to define a group of LUs. The following example shows that when the workstation 9.24.105.220 connects to z/OS, it will be assigned to one of the LUs RA03TN70 through RA03TN75, which means that the workstation 9.24.104.28 can have up to six Telnet sessions when accessing z/OS:

```
LUGROUP LUGRP1
      RA03TN70 RA03TN71 RA03TN72 RA03TN73 RA03TN74 RA03TN75
ENDLUGROUP
LUMAP LUGRP1 9.24.105.220
```

3. Map a group of LUs with a group of IP addresses

In order to associate a group of LUs and a group of IP addresses you have to use LUGROUP/ENDLUGROUP statements to define a group of LUs and IPGROUP/ENDIPGROUP statements to define a group of IP addresses. The following example shows that when any of the workstations on the 9.24.105.0 subnet connect to z/OS, they will be assigned one of the LUs RA03TN01 through RA03TN20:

```
LUGROUP EN1LUG
      RA03TN01..RA03TN20
ENDLUGROUP
IPGROUP EN1IPG 255.255.255.0:9.24.105.0 ENDIPGROUP
LUMAP EN1LUG EN1IPG
```

4. Map an LU name with a host name

In the following example, workstation m238p4rk.itso.ral.ibm.com will always be assigned to an LU name of RA03TN70:

```
LUMAP RA03TN70 m238p4rk.itso.ral.ibm.com
```

5. Map a group of LUs with a host name

In order to associate a group of LUs and a host name, you have to use LUGROUP/ENDLUGROUP statements to define a group of LUs. The following example shows that when the workstation m238p4rk.itso.ral.ibm.com connects to z/OS, it will be assigned to one of the LUs RA03TN01 through RA03TN05, which means that the workstation m238p4rk.itso.ral.ibm.com can have up to five Telnet sessions when accessing z/OS:

```
LUGROUP NTLUG
      RA03TN01 RA03TN02 RA03TN03 RA03TN04 RA03TN05
ENDLUGROUP
LUMAP NTLUG m238p4rk.itso.ral.ibm.com
```

6. Map a group of LUs with a group of host names

In order to associate a group of LUs and a group of IP addresses you have to use LUGROUP/ENDLUGROUP statements to define a group of LUs, and HNGROUP/ENDHNGROUP statements to define a group of IP addresses. The following example shows that when any of the workstations on the itso.ral.ibm.com domain connect to z/OS, they will be assigned one of the LUs RA03TN61 through RA03TN80:

```
LUGROUP TR1LUG
      RA03TN61..RA03TN80
ENDLUGROUP
HNGROUP TR1HNG **.itso.ral.ibm.com ENDHNGROUP
LUMAP TR1LUG TR1HNG
```

As SNA security normally relies on the LU names, it is imperative to have means to ensure the IP addresses using Telnet really are the ones they pretend to be. This requires an adequate network security management.

The LUMAP statement that mapped the LU name to the IP address assumed a static IP address at the end user. But with dynamic IP assignment, this assumption is no longer valid because the DHCP may assign a different IP address to the client every time the client connects to the Telnet server.

LUMAP with host name mapping will help you solve this problem. To do so, DHCP is required to allow clients to specify host names. And DDNS is required as an interface to the DHCP server for host name updates. This solution may avoid specifying LU names in the TN3270E clients.

7. Map a group of LUs with a group of destination IP addresses

In order to associate a group of LUs and a group of destination IP addresses you have to use LUGROUP/ENDLUGROUP statements to define a group of LUs and DESTIPGROUP/ENDDESTIPGROUP statements to define a group of destination IP addresses. Following is an example, mapping LUs RA03TN01 and RA03TN02 to destination IP addresses 1.2.3.5 and 1.2.3.6:

```

LUGROUP LUGRP1
      RA03TN01 RA03TN02
ENDLUGROUP
DESTIPGROUP DSTIPGRP1
  1.2.3.5 1.2.3.6
ENDESTIPGROUP
LUMAP LUGRP1 DSTIPGRP1

```

8. Map a group of LUs with a group of link names

In order to associate a group of LUs and a group of link names, you have to use LUGROUP/ENDLUGROUP statements to define a group of LUs and LINKGROUP/ENDLINKGROUP statements to define a group of link names. Following is an example, mapping LUs RA03TN61 to RA03TN80 to link names LINK1, LINK2 and LINK3:

```

LUGROUP LUGRP2
      RA03TN61..RA03TN80
ENDLUGROUP
LINKGROUP LINKGRP1
  LINK1 LINK2 LINK3
ENDLINKGROUP
LUMAP LUGRP2 LNKGRP1

```

9. Map a group of LUs with a group of user IDs

In order to associate a group of LUs and a group of user IDs, you have to use LUGROUP/ENDLUGROUP statements to define a group of LUs and USERGROUP/ENDUSERGROUP statements to define a group of user IDs. Following is an example, mapping LUs RA03TN01 to RA03TN20 to user IDs MOBL0002, MOBL0003 and MOB1%%C (%% means any character in that position):

```

LUGROUP LUGRP3
      RA03TN01..RA03TN20
ENDLUGROUP
USERGROUP USRGRP1
  MOBL0002 MOBL0003
  MOB1%%C
ENDUSERGROUP
LUMAP LUGRP3 USRGRP1

```

10. This syntax, implemented by OS/390 V2R10, applies to all of the syntax options above. Additionally, it provides a default application with the LOGAPPL functionality in the LUMAP statement.

```

LUMAP RA03TN70 9.24.105.220 SPECIFIC DEFAPPL RA03T LOGAPPL RA03TP70
PRTMAP RA03TP70 9.24.105.220

```

```

LUGROUP LUGRP1
      RA03TN90..RA03TN99
ENDLUGROUP
PRTRGROUP PRTGRP1
      RA03TP90..RA03TP99
ENDPRTGROUP
LUMAP LUGRP1 9.24.105.220 SPECIFIC DEFAPPL RA03N LOGAPPL PRTGRP1
PRTMAP PRTGRP1 9.24.105.220

```

You can associate printers at the same time in the LUMAP syntax with the DEFAPPL parameter. Also you can use them without printer association or without the DEFAPPL parameter.

Application selection

After a client is connected to the Telnet server, an application, a USS message 10 screen, or a network solicitor screen will be sent to the client. Then an application permission check and LU allocation will occur based on a default application name or a user-entered application name. If the application is permitted, an LU-LU session will be established.

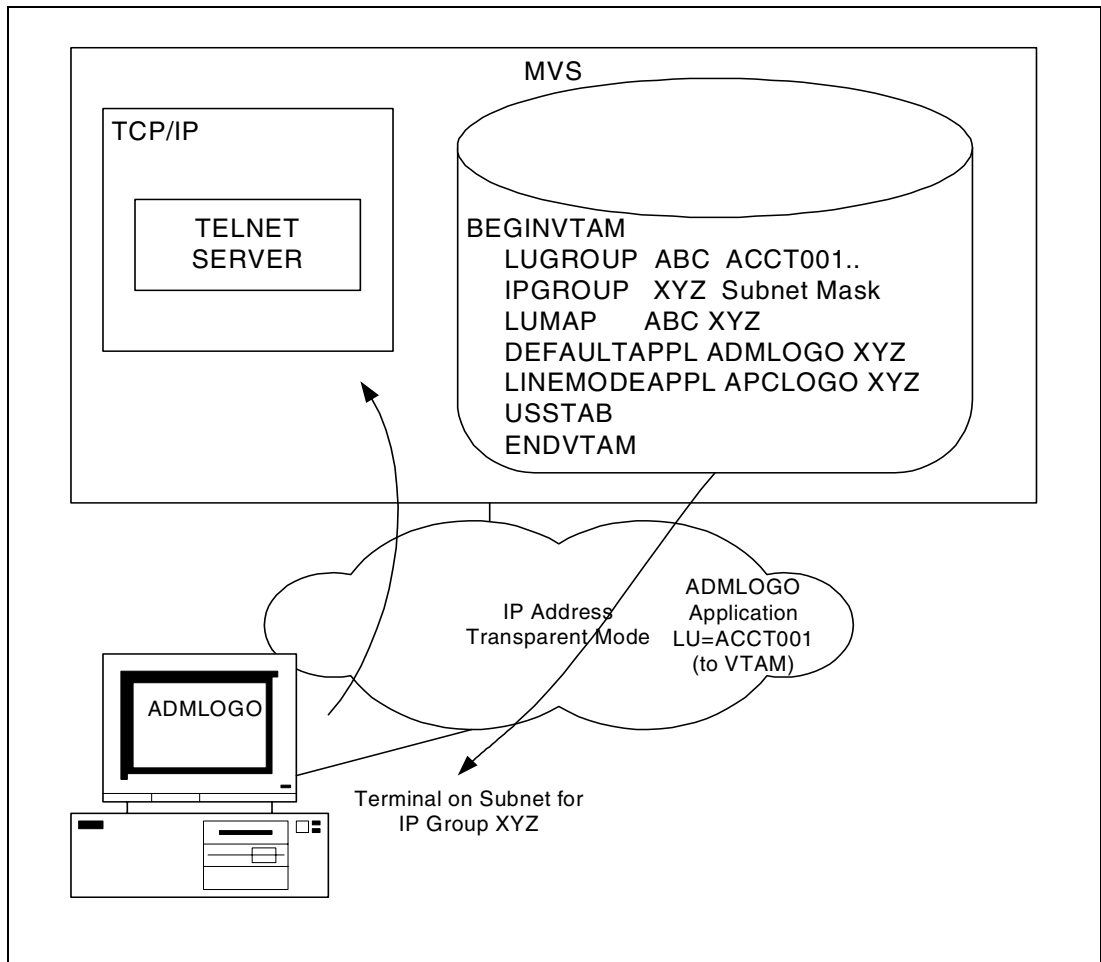


Figure 8-11 IP address to LU mapping

LUMAP statement with DEFAPPL option

You can define an application to an LU or LUGROUP with the LUMAP statement. The DEFAPPL parameter on the LUMAP statement allows a host application to be mapped with an LU name or LUGROUP name instead of using DEFAULTAPPL. The LUMAP-DEFAPPL statement is treated just like DEFAULTAPPL when a client identifier matches the LUMAP statement. The LUMAP-DEFAPPL statement also supports the LOGAPPL, FIRSTONLY, and DEFONLY parameters that are used by DEFAULTAPPL and LINEMODEAPPL. You can see a sample definition in Figure 8-12.

```

;TCPIP.TCPPARMS.R2615(TELN03A) - 01.44
TELNETPARMS
  TKOSPECLU 0
  PORT 23
  WLMCLUSTERNAME TN03 TNRAL TNTSO  ENDWLMCLUSTERNAME
ENDTELNETPARMS

BEGINVTAM
  PORT 23
  ALLOWAPPL *
  MSG07
;*** IP2 TN3270 DEFINITIONS **

; IP2 group includes 9.24.105.220 and 9.24.106.165 ip addresses
IPGROUP IP2
9.24.105.220 9.24.106.165
ENDIPGROUP
  LUGROUP LU2 RA03TN50..RA03TN51 ENDLUGROUP
  PRTGROUP PR2 RA03TP50..RA03TP51 ENDPRTGROUP
; LUTSO group includes lus from RA03TN70 to RA03TN75
LUGROUP LUTSO
  RA03TN70..RA03TN75
ENDLUGROUP
; This mapping maps LUTSO lu group to IP2 ip group and supports direct
; logon to RA03T application for LUTSO group name and members.
LUMAP LUTSO IP2 SPECIFIC DEFAPPL RA03T LOGAPPL

```

Figure 8-12 Sample LUMAP-DEFAPPL definition

LINEMODEAPPL and DEFAULTAPPL statements

You have the following options for selecting an initial application or a USS message 10 screen:

1. You can use the LINEMODEAPPL statement to specify an initial application to which to connect when a Telnet client establishes a line mode connection.
2. You can use the DEFAULTAPPL statement to specify an initial application to which to connect when a Telnet client establishes a 3270 full-screen connection.
3. The LOGAPPL function can be coded on DEFFAULTAPPL and LINEMODEAPPL. The LOGAPPL function keeps the Telnet LU active if a Request Session fails because the host application is not active. In addition, VTAM remembers the attempted Request Session and will initiate a session request to the Telnet LU on behalf of the host application when the application becomes active. When the Request Session fails, Telnet sends the client a solicitor panel or USSMSG7 screen.
4. You can use the USSTCP statement to specify a USS message 10 screen to show when a Telnet client establishes a 3270 connection.
5. You can use the INTERPTCP statement to specify a customized interpret table, which is used to interpret incoming USS commands before the USS command processor is invoked.
6. You can specify none of the above and get the Telnet solicitor screen.

Telnet uses the sequence shown in Table 8-4 on page 197 to select an application or a USS message 10. If neither an application nor a USS message 10 can be mapped to the client, then the default Telnet solicitor screen is displayed to get an application from the user.

If the USSTCP statement is specified but the USS table module cannot be found in the system's linklist or TCPIP STEPLIB, a default USS table (EZBTPUST) will be used.

RESTRICTAPPL and ALLOWAPPL statements

When an application name is selected, an LU is allocated based on the client identifier selection and application status (restricted, allowed) using the following criteria:

1. Checks if it is a restricted application (RESTRICTAPPL).

The RESTRICTAPPL statement restricts access to the specified application. This statement must be followed by a USER statement for each user who is authorized to use the application. When a restricted application is selected, users are prompted to identify themselves with a user ID and a password. RACF or equivalent security program is used to validate the password.

The USER statement specifies a user who is authorized to access the application specified by the RESTRICTAPPL statement. Only one user ID is specified on a USER statement, but multiple USER statements can be used to specify multiple user ID definitions. This statement must follow the RESTRICTAPPL statement. For example:

```
RESTRICTAPPL RAKAA
  USER user23
  USER user24
```

You can use LU statements in the RESTRICTAPPL statement to map an LU name and further restrict access to the application, but it complicates administration and may fail with TN3270E clients if not configured properly.

2. Checks if it is an allowed application (ALLOWAPPL).

You can use LU statements in the ALLOWAPPL statement to map an LU name, but the LUMAP statement is the preferred method of allocating LUs. Since OS/390 V2R10, Telnet Server supports LUGROUP name usage for ALLOWAPPL statement. The ALLOWAPPL statement will be used just to specify applications defined in the USS message 10 as applicable.

3. Checks if it is a default application (LINEMODEAPPL, DEFAULTAPPL).

DISCONNECTABLE and QSESSION parameters

On the ALLOWAPPL and RESTRICTAPPL, the DISCONNECTABLE and QSESSION parameters may be specified.

- ▶ DISCONNECTABLE allows applications to be disconnected rather than logged off. This may be of value to VM and NetView/Access Services users. For example, you will be able to *reconnect* to the application before its timeout.
- ▶ QSESSION allows you to select individually which VTAM applications queue their sessions when performing a CLSDST/PASS. CLSDST/PASS information is stored up to 10 sessions deep. QSESSION is only needed on the first application entered, for example NetView/Access Services.

Note: The LU is selected during negotiation for TN3270E connections before the application is chosen. Care must be taken to ensure that an LU name chosen based solely on LU mapping during TN3270E negotiation is not later rejected because of LU statements on the RESTRICTAPPL and ALLOWAPPL statements.

Generic and specific device pools for LU names

Server LU device name pools (terminal or printer) can be defined as either generic or specific. Specific pools are used to satisfy requests from clients that request a particular (specific) device name.

Generic pool LUs are used to satisfy connection requests where no device name is requested. If a specific request fails to match a device name in the specific LU pool, the generic pool will be searched. If again no match is found, the connection will be dropped by the server.

TN3270E recognizes four kinds of device pools: generic and specific terminals (LUMAP) and generic and specific printers (PRTMAP). Here are the rules for terminal and printer connections:

- ▶ The client can connect without a device name request and let the server select the terminal LU for the client. This function is also available for printer connections. The server selects a name and sends it to the client. If the client agrees, negotiation continues. Otherwise, the connection is dropped.
- ▶ The client can request a specific device name during negotiation. If the server PROFILE definitions allow the use of this name for the particular client identifier, that name is reserved for the client and negotiations continue. First the specific group is checked; if no match is found, the generic group is checked.

The LUGROUP or PRTGROUP statement with the SPECIFIC parameter configures the specific pool. The following is an example:

```
LUMAP SPECLX SPECIP SPECIFIC
```

The LUGROUP or PRTGROUP statement with the GENERIC parameter configures the generic pool. The following is an example:

```
LUMAP GENLX GENIP GENERIC
```

GENERIC is the default for all LUMAP and PRTMAP statements.

Printer definition

As part of the TN3270E protocol the Telnet server supports 3287-type printers. The printers are basically defined in the same way as display LUs. Just use PRTGROUP and PRTMAP instead of LUGROUP and LUMAP.

The client can also ask for a printer session that is associated with an existing display session. The server must know through profile statements which printer LU is associated with which display LU. There must be a one-to-one mapping of display LUs to printer LUs. This mapping can be either a single display LU to a single printer LU or a display LU group to a printer LU group with each group having the same number of LUs. The association is done using the LUMAP statement. The last parameter on the LUMAP statement is the printer LU name or printer LU group name that is associated with the display LU name or display LU group in the LUMAP statement.

```
LUMAP SPECLX SPECIP GENERIC PRTL
```

Printer sessions that are not associated require either the PRTMAP/PRTGROUP statements for direct printer connections or the LUMAP statement for associated printer sessions. You can use specific or generic mapping. For an explanation of specific mapping see “Generic and specific device pools for LU names” on page 203.

LU mapping samples

Figure 8-13 is a sample Telnet PROFILE configuration. The markers in this figure are related to those in Table 8-4 on page 197 and to the explanations that follow.

```

TELNETPARMS
  PORT 23
  INACTIVE 600 TIMEMARK 600 SCANINTERVAL 120
  SMFINIT STD SMFTERM STD
  WLMCLUSTERNAME TNRAL ENDWLMCLUSTERNAME
ENDTELNETPARMS
TELNETPARMS
  SECUREPORT 223 KEYRING MVS TCP.Telnet.KYR
  INACTIVE 600 TIMEMARK 600 SCANINTERVAL 120
  SMFINIT STD SMFTERM STD
  WLMCLUSTERNAME TNRALSSL3 ENDWLMCLUSTERNAME
ENDTELNETPARMS

BEGINVTAM
  PORT 23 223

  DEFAULTLUS
    RA&SYSCLONE.TN01..RA&SYSCLONE.TN50           A
  ENDDEFAULTLUS

  LUGROUP SPECLU
    RA&SYSCLONE.TN91..RA&SYSCLONE.TN99           B
  ENDLUGROUP
  LUMAP SPECLU 9.24.104.201

  IPGROUP SPECIP
    9.24.104.191 9.24.104.192 9.24.104.193 9.24.104.194 9.24.104.195
    9.24.104.196 9.24.104.197 9.24.104.198 9.24.104.199
  ENDIPGROUP

  LUGROUP SPECLX
    RA&SYSCLONE.TN71..RA&SYSCLONE.TN75           C
  ENDLUGROUP

PRTGROUP PRINTERS
  RA&SYSCLONE.TPR1..RA&SYSCLONE.TPR5
  ENDPRTGROUP           ; printers for specific mapping and
                        ; printer to LU association
  LUMAP SPECLX SPECIP SPECIFIC PRINTERS

PRTGROUP PRINTERG
  RA&SYSCLONE.TPR6..RA&SYSCLONE.TPR9           D
  ENDPRTGROUP
  PRTMAP PRINTERG SPECIP

LUGROUP RISCLUG
  RA&SYSCLONE.TN82 RA&SYSCLONE.TN83 RA&SYSCLONE.TN84 E
  RA&SYSCLONE.TN85 RA&SYSCLONE.TN86
  ENDLUGROUP
  LUMAP RISCLUG 9.24.104.28

LUGROUP TR1LUG
  RA&SYSCLONE.TN61..RA&SYSCLONE.TN80           F
  ENDLUGROUP
  IPGROUP TR1IPG 255.255.255.0:9.24.104.0 ENDIPGROUP
  LUMAP TR1LUG TR1IPG

```

Figure 8-13 Example of LU mapping (part 1)

```

LUGROUP NTLUG
  RA&SYSCclone.TN51..RA&SYSCclone.TN59
ENDLUGROUP
HNGROUP HNG
  **.ral.ibm.com
ENDHNGROUP
LUMAP NTLUG HNG SPECIFIC

LUMAP RA&SYSCclone.TN81 9.67.32.10
LUMAP RA&SYSCclone.TN60 wtr05246.itso.ral.ibm.com

USSTCP TELNUSS
USSTCP TELNUST 9.24.104.28
DEFAULTAPPL RAKAA TRIIPG
LINEMODEAPPL RA&SYSCclone.T ICP1

ALLOWAPPL RA*
ALLOWAPPL AD*
ALLOWAPPL A2*
ALLOWAPPL FD*
ALLOWAPPL X6*
ALLOWAPPL X7*
ENDVTAM

```

Figure 8-14 Example of LU mapping (part 2)

Notes:

A Default LU name selection

You can use DEFAULTLUS/ENDDEFAULTLUS to define a universal pool of LUs. These LUs are used when the application selected by the user does not meet the criteria for an LU, as defined in the ALLOWAPPL, RESTRICTAPPL, or LUMAP statements. When defining this universal pool of LUs, avoid coding MAPPED LU NAMES in the DEFAULTLUS/ENDDEFAULTLUS statement also. It prevents the pool of mapped LUs from being exhausted.

B LU name selection based on the IP address

The workstation with the IP address 9.67.32.10 connecting to z/OS will be mapped with RA39TN81. &SYSCclone. variable is 39 on this system. Also the workstation with the IP address 9.24.104.201 will be mapped with the LU from RA39TN91 through RA39TN99.

C Terminal LU to printer LU association

Users with IP addresses defined in SPECIP can get an LU name from the SPECLX group and later an associated printer; they can also directly request a specific printer LU.

To be of use, these printers should also be defined in the VTAM application.

D Printer LU selection based on the IP address

All users with addresses defined in SPECIP can get printer sessions.

E Initial screen and LU name selection based on the IP address

The workstation with the IP address 9.24.104.28 connecting to z/OS in transparent mode will get USS message 10 from the USS table TELNUST. Also, the first LU name available in the RISCLUG LU group from RA39TN82 through RA39TN86 will be assigned.

f Initial screen and LU name selection based on the IP addresses group

The workstations on the 9.24.104.0 subnet (except the workstation with the IP address 9.24.104.28) is assigned to one of the LUs RA39TN61 through RA39TN80 with the full-screen application called RAKAA.

g LU name selection based on the host names group

The workstations with the host name that matches the wild card `**.*.ral.ibm.com` is assigned to one of the LUs RA39TN51 through RA39TN59.

h LU name selection on the basis of host name

The workstation with the host name `wtr05246.itso.ral.ibm.com` connecting to z/OS will be mapped with RA39TN60.

i Initial screen selection with no option specified

If TCP/IP does not find an application or USS message 10 based on the client's host name, IP address, or link name, it tries to find it with no optional parameter. In this case, TCP/IP finds TELNUSS as a universal USS message 10. This search sequence is shown in Table 8-4 on page 197.

j Initial screen selection based on the link name

Any users of the ICP1 network interface will use the RA39T line mode application when connecting to z/OS in line mode.

8.2.5 USS messages

If you do not want to use the default Telnet solicitor logon panel, you can implement a customized Unformatted System Services (USS) screen as the Telnet logon screen. Telnet USSMSG tables can be used to send messages to the client and process commands from the client. Telnet USSMSG tables provide you with a way to emulate VTAM USSMSG processing. For more details on how to implement these USS messages, see *z/OS V1R2.0 Communications Server IP Configuration Guide, SC31-8775*.

USS services are used for 3270 devices only. You may include information about logon errors on the message 10 screen by USS message 7 support.

USS message support has been improved to bring TCP/IP more in line with VTAM. End users can enter APPLID, LOGMODE and DATA parameters. Date and time are additional substitution variables on USSMSG. Functional support is included in IBMTEST and LOGOFF commands. Several message types are also included.

You can use the USSTCP mapping rules to associate a customized USS message 10 screen with an IP address, a group of IP addresses, a host name, a group of host names, or a link name (for example, a client identifier).

When you create a USS table for the TCP/IP environment, you must be aware of certain rules and restrictions:

1. The USS table load module should be accessible to the TCP/IP and Telnet address space. You can ensure that by placing the load library holding the USS table in the STEPLIB concatenation of the TCP/IP address space. See **f** in Figure 8-15.

```

//T03ATCP  PROC PARMS='CTRACE(CTIEZB01)',
// XPARM=' ENVAR("RESOLVER_CONFIG=//TCP.TCPPARMS(TDATA03A) ") '
:
:
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//      DD DSN=TCPIP.ITSC.LINKLIB,DISP=SHR
:
:

```

Figure 8-15 TCP/IP address space STEPLIB for USS table

2. You may reuse any USS table already in use in your VTAM environment, as long as it is one that has been coded for *non-SNA* terminals.
 - ▶ TN3270 and TN3270E clients understand only the 3270 data stream and *non-SNA* commands with USS messages. Except for the **READ MODIFIED ALL** command, the commands valid for an SNA environment are also valid for the non-SNA locally attached environment. For example, the **ERASE/WRITE** command (Figure 8-17 on page 210) is valid in both SNA and non-SNA environment, but the command code is different. Thus we used an x'05' command code instead of an x'F5' command code to define the **ERASE/WRITE** command in Figure 8-17 on page 210.
 - ▶ To activate Telnet's character string substitution you must use the LUNAME or SCAN subparameter of the BUFFER parameter on the USSMSG macro.
 - ▶ You may use the TEXT instead of the BUFFER parameter in USS messages.
3. Following are the character strings supported by character string substitution:

Table 8-6 Character strings supported

Character string	Explanation
@ @LUNAME	Server (SLU) Name
@ @ @ @RUNAME	Failing RU name
@ @ @SENSE	Sense Code
@ @ @ @DATE	Current Date
@ @ @ @TIME	Current Time
@ @ @ @ @ @ @ @IPADDR	Client IP Address
@ @PRT	Client Port Number

4. The USS table can be changed, re-assembled, and put back into the same library ready to be loaded when the next **OBEYFILE** command processes the Telnet profile. The end result will be existing connections continue to use the old table but any new connections will use the newly assembled table.
5. LU names are reserved for only TN3270E connections when the user connects. TN3270 and line mode connections do not reserve an LU. When the application name is entered, TN3270E will verify that the reserved LU is valid for the application chosen and will open the ACB. TN3270 and line mode will select an LU based on CLID to object mapping and then open the ACB.

Note: Since ACB will not be opened until the application name is entered, an SNA application may not be able to acquire the Telnet display session. Printer session is an exception.

- TCP/IP supplies a default USS table named EZBTPUST. You will find the load module in the SEZALINK data set. The source code is in the SEZAINST and has the same name.
If the USSTCP statement is specified but the USS table module cannot be found in the system's linklist or TCP/IP's STEPLIB, the following default USS table (EZBTPUST) will be used, as shown in Figure 8-16.

```
USSMSG10: Enter: LOGON APPLID() LOGMODE() DATA()  
Port: 01079          Date: 07/09/01  
IPADDR: 9.24.105.220 Time: 11:05:22
```

Figure 8-16 Default USS table

- Logon errors are handled by the MSG07 parm and logoff from applications is handled by the LUSESSIONPEND parm. With MSG07 coded, logon errors will result in MSG07 or MSG04 being returned to the client. Without MSG07 coded, logon errors result in connection drops. With LUSESSIONPEND coded, logoff from the application will result in a new MSG10 or solicitor screen being sent to the client or the default application redriven. Without LUSESSIONPEND coded, logoffs from applications will result in connection drops.

Sample USS table coding and results

Figure 8-17 is a sample source USS table that was used for testing the Telnet USS messages 7 and 10.

```

USST      TITLE 'USSTAB FOR 5203TELN WITH NEW APPLICATIONS'
USSITSO  USSTAB  FORMAT=DYNAMIC
*
TS003    USSCMD  CMD=TS003,REP=LOGON,FORMAT=BAL
          USSPARM PARM=APPLID,DEFAULT=RA3AT
          USSPARM PARM=P1,REP=DATA
          USSPARM PARM=P2,REP=LOGMODE
          ..
          ..

IMS      USSCMD  CMD=IMS,REP=LOGON,FORMAT=BAL
          USSPARM PARM=APPLID,DEFAULT=T18AIMS
*
MESSAGE7 USSMSG  MSG=7,BUFFER=(MSG07,SCAN)
          DS      OF
MSG10    DC      AL2(MSG07E-MSG07S)
MSG10S   DC      X'05C21D70114040',C'MSG07 OE/390 (03)'
          ..
          ..
          Messages 7 and 10 are
          the same in this USS table,
          except for the number in
          line 1 column 4 and 5.
MSG07E   EQU      *
*
MESSAG10 USSMSG  MSG=10,BUFFER=(MSG10,SCAN)
          DS      OF
MSG10    DC      AL2(MSG10E-MSG10S)
MSG10S   DC      2X'05C21D70114040',C'MSG10 OE/390 (03)'
          DC      X'11C150'
          DC      C'Raleigh - International Technical Support OrganizationC
          DC      - ITS0 - ITS01'
          DC      X'11C2F1'
          DC      C'System OE/390 (03)'
          DC      X'11C3F2'
          DC      C' '
          DC      X'114AC9',C'Enter:'
          DC      X'114A50',X'1D4013',CL41' ',X'1D40'
          DC      X'114BD9',C' '
          DC      X'114CE9',C'TS003 userID - TSO on MVS03'
          DC      X'114DF9',C'TS028 userID - TSO on MVS28'
          DC      X'114FC9',C'CICS - CICS on MVS03'
          DC      X'1150D9',C'NVAS20 - NetView Access on MVS20'
          DC      X'11D1E9',C'IMS - IMS on MVS18'
          DC      X'11D2F9',C'SYS6 - RALYDPD6'
          DC      X'11D4C9',C' '
          DC      X'11D5D9',C' '
          DC      X'11D660',C'Your IP Address: @@@@IPADDR'
          DC      X'11D7D3',C'Your Telnet Port: @@PRT '
          DC      X'11D7F0',C'-----'
          DC      X'11D85E',C'-----Last command: @@@RUNAME'
          DC      X'11D940',C'LU: @@LUNAME Sense Code: @@@SENSE'
          DC      X'11D9F3',C'Date: @@@DATE Time: @@@@TIME'
MSG10E   EQU      *
END      USSEND
          END

```

Figure 8-17 USSMSG7 and USSMSG10 sample

Figure 8-18 displays the USS message 10 screen that results from the coding in Figure 8-17.

```
MSG10 OE/390 (03)
Raleigh - International Technical Support Organization - ITS0 - ITS01
                System OE/390 (03)

Enter:

TS003 userID - TSO on MVS03
TS028 userID - TSO on MVS28
TS039 userID - TSO on MVS39
CICS          - TCP/IP CICS on MVS18
NVAS20        - NetView Access on MVS20
IMS           - TCP/IP IMS on MVS18
SYS6          - RALYDPD6

Your IP Address: 9.24.105.220          Your Telnet Port: 01318
-----Last command:
LU: RA03TN50      Sense Code:          Date: 10/09/01 Time: 11:18:23
```

Figure 8-18 USSMSG10: initial view

In the following screen you can also observe the contents of the USS message 7 keywords after the user has entered the USSCMD NVAS20 to request a cross-domain application that is currently not reachable. Notice the sense code of 087D0001, which indicates the application is unknown to VTAM.

```
MSG07 OE/390 (03)
Raleigh - International Technical Support Organization - ITS0 - ITS01
                System OE/390 (03)

Enter:

TS003 userID - TSO on MVS03
TS028 userID - TSO on MVS28
TS039 userID - TSO on MVS39
CICS          - TCP/IP CICS on MVS18
NVAS20        - NetView Access on MVS20
IMS           - TCP/IP IMS on MVS18
SYS6          - RALYDPD6

Your IP Address: 9.24.105.220          Your Telnet Port: 01318
-----Last command: REQSESS
LU: RA03TN50      Sense Code: 087D0001  Date: 10/09/01 Time: 11:20:12
```

Figure 8-19 USSMSG07: USSMSG7 contents after unsuccessful logon

8.2.6 Using translation tables

Two types of translation tables are used by OS/390 V2R5 IP and later. SBCS are used for single-byte characters. DBCS translation tables are used for converting double-byte characters. DBCS are required for character sets such as Japanese Kanji.

For more information on how to use translation, see:

- ▶ Chapter 6, “National Language Support (NLS)” on page 139
- ▶ *z/OS V1R2.0 Communications Server IP Configuration Guide*, SC31-8775
- ▶ *z/OS V1R2.0 Communications Server IP User's Guide and Commands*, SC31-8780

8.3 Operating the Telnet environment

The operator can use **VARY** commands to control Telnet and display commands to view profile, connection and port information. These commands are described in *z/OS V1R2.0 Communications Server IP System Administrator's Commands*, SC31-8781.

8.3.1 Telnet VARY commands

Telnet **VARY** commands give the operator control over stopping and starting Telnet and allowing clients to connect.

The combination of the **STOP**, **QUIESCE**, **RESUME**, and **OBEYFILE** commands gives the operator complete control over when to stop and start Telnet and when to allow end users to connect. To help manage commands related to multiple ports, **VARY** and **DISPLAY** commands for the profile, connection, and port categories support a **PORT** keyword. Telnet **VARY** commands include:

- ▶ **VARY QUIESCE** causes the port not to accept any new Telnet connections. The existing connections are not affected.
- ▶ **VARY RESUME** to end the QUIESCE state and accept new connections.
- ▶ **VARY STOP** to end all the connections to the Telnet port and close the port.
- ▶ Start or restart a port using the **VARY OBEYFILE** command (to update the Telnet PROFILE). Using this command, you can stop Telnet activity on one port and begin activity on a new port without stopping the TCP/IP stack. You can also start activity on new ports without stopping activity on existing ports.
- ▶ **VARY ACTIVATE** and **INACTIVATE** to activate and deactivate LUs from the Telnet server's perspective. If an LU is already in use, the **INACT** command will fail.

Note: These commands are Telnet commands and have no effect on VTAM's LU status.

- ▶ **VARY OBEYFILE** is used to update the Telnet profile or to restart a Telnet port if the **VARY STOP** command has been issued.
- ▶ When the **OBEYFILE** command is issued, the existing profile becomes non-current. A new current profile is then used to serve all new connections. The non-current profile keeps serving its old connections.

Note: If you **QUIESCE** or **STOP** a port defined with the **WLMCLUSTERNAME** statement, the **QUIESCE** and **STOP** commands will deregister the Telnet server from WLM. This will result in host unknown type messages for clients that attempt to connect to the Telnet server using the **WLMCLUSTERNAME** in a DNS/WLM sysplex. **RESUME** command will re-register the Telnet server with WLM. If there is more than one port defined with **WLMCLUSTERNAME**, the deregister will only occur when you stop the last WLM port.

VARY ACT command

The **VARY ACT** command changes the availability status of a VTAM LU for Telnet server usage. **ACT** enables the specified LU to be a candidate to represent a Telnet client.

```
V TCPIP,T39ATCP,T,ACT,RA39TN77
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,T,ACT,RA39TN77
EZZ6038I TELNET COMMAND ACT RA39TN77 COMPLETE
```

Figure 8-20 Telnet VARY ACT command

Note: The LUs must also be activated in VTAM or the openACB will fail and the LUs will be inactivated again.

VARY INACT command

The **VARY INACT** command changes the availability status of a VTAM LU for Telnet server usage. **INACT** disables the LU as a candidate to represent a Telnet client.

```
V TCPIP,TCPIPA,T,INACT,RA03TN51
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,T,INACT,RA03TN51
EZZ6038I TELNET COMMAND INACT RA03TN51 COMPLETE
```

Figure 8-21 Telnet VARY INACT command

But you cannot use the **VARY INACT** command for used LUs. You will get an error message as in Figure 8-22.

```
V TCPIP,TCPIPA,T,INACT,RA03TN51
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,T,INACT,RA03TN51
EZZ6039I TELNET COMMAND INACT RA03TN51 FAILED, RC = 3001
```

Figure 8-22 Telnet VARY INACT command for used LUs

VARY OBEYFILE command

The **VARY OBEYFILE** command is not a Telnet command, but it is used to update the Telnet profile or to restart a Telnet port.

Update profiles do not need to contain statements for all active ports. If the profile does not contain a **TELNETPARMS** block or a **BEGINVTAM** block for a port that is currently active, the port remains active with its current profile.

```

V TCPIP,TCPIPA,0,TCPIP.TCPPARMS.R2615(TELN03A)
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,0,TCPIP.TCPPARMS.R2615(TELN03A)
EZZ0300I OPENED OBEYFILE FILE 'TCPIP.TCPPARMS.R2615(TELN03A)'
EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIP.TCPPARMS.R2615(TELN03A)'
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIP.TCPPARMS.R2615(TELN03A)'
EZZ0053I COMMAND VARY OBEY COMPLETED SUCCESSFULLY
EZZ0400I TELNET/VTAM (SECONDPASS) BEGINNINGFORFILE: 'TCPIP.TCPPARMS.R2615(TELN03A)'
EZZ0201I NETWORK REFERENCE IN DATABASE REPLACED BY THIS ONE ON LINE 112
EZZ6018I TELNET PROFILE UPDATE COMPLETE FOR PORT 23
EZZ6018I TELNET PROFILE UPDATE COMPLETE FOR PORT 6623
EZZ0403I TELNET/VTAM (SECOND PASS) COMPLETE FOR FILE: //'TCPIP.TCPPARMS.R2615(TELN03A)'

```

Figure 8-23 OBEYFILE command with Telnet

Note: The processing of the Telnet server begins with the EZZ0400I message and ends with the EZZ0403I message.

VARY QUIESCE command

The **VARY QUIESCE** command causes the specified port not to accept any new Telnet client connections. Currently established connections continue to be serviced. The current profile is retained as long as an **OBEYFILE** command does not create a new profile.

```

V TCPIP,TCPIPA,T,QUIESCE,PORT=6623
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,T,QUIESCE,PORT=6623
EZZ6003I TELNET QUIESCED ON PORT 6623

```

Figure 8-24 Telnet VARY QUIESCE command

VARY RESUME command

The **VARY RESUME** command causes the currently QUIESCED port to begin accepting new Telnet client connections again using either the existing profile or a new profile. If a **VARY OBEYFILE** command has been issued while the port was QUIESCED, a new profile will be used.

```

V TCPIP,TCPIPA,T,QUIESCE,PORT=6623
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,T,RESUME,PORT=6623
EZZ6003I TELNET RESUMED ON PORT 6623

```

Figure 8-25 Telnet VARY RESUME command

VARY STOP command

The **VARY STOP** command ends the port connection and all active connections. After the **VARY STOP** command, the command processor is still active and you can use a **VARY OBEYFILE** command to activate a Telnet port using the Telnet configuration parameters.

```

V TCPIP,TCPIPA,T,STOP,PORT=6623
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,T,STOP,PORT=6623
EZZ6010I TELNET SERVER ENDED FOR PORT 6623

```

Figure 8-26 Telnet VARY STOP command

8.3.2 Telnet DISPLAY commands

The profile displays have been streamlined to match the mapping of objects to client identifiers concept. Several profile displays are no longer supported and have been replaced by Object or ClientID displays. The old commands are still accepted, but the output is in the Object or ClientID format. The same, or more, information is provided. The following displays are *no longer* supported and internally generate the specified Object or ClientID:

- ▶ Appl
- ▶ Defaults
- ▶ IpGroup
- ▶ HnGroup
- ▶ Linkname
- ▶ LuGroup
- ▶ LuMap
- ▶ WhereUsed

Telnet **DISPLAY** commands support multiple dynamic profiles in Telnet. The **DISPLAY** commands are used to supply the operator with information concerning:

- ▶ Profiles
- ▶ Connections
- ▶ Port
- ▶ Server

The new DISPLAY commands available are:

- ▶ **Display Telnet ClientID**
- ▶ **Display Telnet DEVICETYPE**
- ▶ **Display Telnet OBJECT**
- ▶ **Display Telnet PROFILE**
- ▶ **Display Telnet Group Connection**
- ▶ **Display Telnet Group WLM**
- ▶ **Display Telnet Group INACTLUS**

For a complete description of each command, please refer to *z/OS V1R2.0 Communications Server IP System Administrator's Commands*, SC31-8781.

8.3.3 VTAM display commands

The following examples show the output of the VTAM application program major node, as in Figure 8-27, and a minor node, as in Figure 8-28. ACT/S means VTAM ACB has been opened and an LU-LU session has been established.

```
D NET, ID=TELAPPL, E
IST097I DISPLAY ACCEPTED
IST075I NAME = TELAPPL, TYPE = APPL SEGMENT 178
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
IST360I APPLICATIONS:
IST080I RA03TN?? CONCT      RA03TN51 ACT/S      RA03TP?? CONCT
IST314I END
```

Figure 8-27 Telnet major node display

OS/390 V2R5 and later has added a function to report the IP addresses associated with OS/390 Telnet server clients and client IP addresses reported by dependent LUs acting as Telnet servers. The message shows the IP address of the TN3270 client in dotted decimal form in the IST1669I message.

```

D NET, ID=RA03TN51, E
IST097I DISPLAY ACCEPTED
IST075I NAME = USIBMRA.RA03TN51, TYPE = DYNAMIC APPL 186
IST486I STATUS= ACT/S, DESIRED STATE= ACTIV
IST1447I REGISTRATION TYPE = CDSERV
IST1629I MODSRCH = NEVER
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=ISTINCLM USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=***NA*** USS LANGTAB=***NA***
IST1632I VPACING = 7
IST597I CAPABILITY-PLU ENABLED ,SLU ENABLED ,SESSION LIMIT 00000001
IST231I APPL MAJOR NODE = TELAPPL
IST1425I DEFINED USING MODEL RA03TN??
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = TCPIPA, STEPNAME = TCPIPA, DSPNAME = IST7512E
IST228I ENCRYPTION = OPTIONAL , TYPE = DES
IST1563I CKEYNAME = RA03TN51 CKEY = PRIMARY CERTIFY = NO
IST1552I MAC = NONE MACTYPE = NONE
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = 0 MAXIMUM = 0
IST1669I IPADDR..PORT 9.24.105.220..1078
IST171I ACTIVE SESSIONS = 0000000001, SESSION REQUESTS = 0000000000
IST206I SESSIONS:
IST634I NAME      STATUS      SID      SEND RECV VR TP NETID
IST635I RA03T03  ACTIV-P    C7335B7CBE2C4935 0581 05D5      USIBMRA
IST314I END

```

Figure 8-28 Telnet application LU display

There is a **DISPLAY IDTYPE - IPADDR**. The operator should issue this **DISPLAY ID IDTYPE**, with input **ID=i.j.k.l** which is TN3270 client IP address in dotted decimal form, and if there is an associated z/OS Telnet server APPL or LU minor resource name, VTAM will output it. An example is shown in Figure 8-29.

```

D NET, IDTYPE=IPADDR, ID=9.24.105.220
IST097I DISPLAY ACCEPTED
IST1668I LUNAME      IPADDR..PORT 215
IST1670I USIBMRA.RA03TN50  9.24.105.220..1956
IST1670I USIBMRA.RA03TN51  9.24.105.220..1078
IST314I END

```

Figure 8-29 VTAM IPADDR display with multiple connections

In this example, a remote Telnet client at the one IP address has multiple TN3270 connections, to one TN3270 server. For this case, since there are multiple LUs, only a summary is presented. You may issue the **D NET, ID=luname, E** command to get detailed information.


```

D NET, IDTYPE=IPADDR, ID=9.24.105.220
IST097I DISPLAY ACCEPTED
IST075I NAME = USIBMRA.RA03TN51, TYPE = DYNAMIC APPL 232
IST486I STATUS= ACT/S, DESIRED STATE= ACTIV
IST1447I REGISTRATION TYPE = CDSERVR
IST599I REAL NAME = USIBMRA.RA03TN51
IST1629I MODSRCH = NEVER
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=ISTINCLM USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=***NA*** USS LANGTAB=***NA***
IST1632I VPACING = 7
IST597I CAPABILITY-PLU ENABLED ,SLU ENABLED ,SESSION LIMIT 00000001
IST231I APPL MAJOR NODE = TELAPPL
IST1425I DEFINED USING MODEL RA03TN??
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = TCPIPA, STEPNAME = TCPIPA, DSPNAME = IST7512E
IST228I ENCRYPTION = OPTIONAL , TYPE = DES
IST1563I CKEYNAME = RA03TN51 CKEY = PRIMARY CERTIFY = NO
IST1552I MAC = NONE MACTYPE = NONE
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = 0 MAXIMUM = 0
IST1669I IPADDR..PORT 9.24.105.220..1078
IST171I ACTIVE SESSIONS = 0000000001, SESSION REQUESTS = 0000000000
IST314I END

```

Figure 8-30 VTAM IPADDR display with a single connection

If there is a single connection found, detailed information will be displayed. Note that the port 1078 is returned as output on the message but was not required as command input.

```

D NET, ID=RA03TN71, E
IST097I DISPLAY ACCEPTED
IST075I NAME = USIBMRA.RA03TN71, TYPE = DYNAMIC APPL 242
IST486I STATUS= ACT/S, DESIRED STATE= ACTIV
IST1447I REGISTRATION TYPE = CDSERVR
IST1629I MODSRCH = NEVER
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=ISTINCLM USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=***NA*** USS LANGTAB=***NA***
IST1632I VPACING = 7
IST597I CAPABILITY-PLU ENABLED ,SLU ENABLED ,SESSION LIMIT 00000001
IST231I APPL MAJOR NODE = TELAPPL
IST1425I DEFINED USING MODEL RA03TN??
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = TCPIPA, STEPNAME = TCPIPA, DSPNAME = IST34C54
IST228I ENCRYPTION = OPTIONAL , TYPE = DES
IST1563I CKEYNAME = RA03TN71 CKEY = PRIMARY CERTIFY = NO
IST1552I MAC = NONE MACTYPE = NONE
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = 0 MAXIMUM = 0
IST1727I DNS NAME: M238P4RK.ITSO.RAL.IBM.COM
IST1669I IPADDR..PORT 9.24.106.165..1241
IST1131I DEVICE = LU - CONTROLLING LU= RA03T
IST171I ACTIVE SESSIONS = 0000000001, SESSION REQUESTS = 0000000000
IST206I SESSIONS:
IST634I NAME STATUS SID SEND RECVR TP NETID
IST635I RA03T12 ACTIV-P C7335B7CBBBD32F31 0000 0002 USIBMRA

```

Figure 8-31 VTAM LU name display for the client defined with DNS name and LOGAPPL parameter

Figure 8-31 shows the VTAM LU name display defined with the HOSTNAME and LOGAPPL parameters. With OS/390 V2R8 and later releases, the VTAM LU display command shows the client's DNS name. OS/390 V2R10 and later supports the LOGAPPL parameter and VTAM displays this application as the CONTROLLING LU. In Figure 8-32 you can see the client TCPIP profile that is used for the display above.

```

TELNETPARMS
  TKOSPECLU 0
  PORT 23
  WLMCLUSTERNAME TN03 TNRAL TNTSO ENDWLMCLUSTERNAME
ENDTELNETPARMS
BEGINVTAM
  PORT 23 6623 7723 8823 9923
  ALLOWAPPL *
; LUTSO group includes lus from RA03TN70 to RA03TN75
  LUGROUP LUTSO
    RA03TN70..RA03TN75
  ENDLUGROUP
; This mapping maps LUTSO lu group to m238p4rk.itso.ral.ibm.com
; hostname and supports direct logon to RA03T application for
; LUTSO group name and members.
  LUMAP LUTSO m238p4rk.itso.ral.ibm.com
    SPECIFIC DEFAPPL RA03T LOGAPPL

```

Figure 8-32 Example TCP/IP profile statements for clients configured with HOSTNAME and LOGAPPL

8.4 Problem determination

This section describes some z/OS CS IP problem determination tools.

8.4.1 Telnet DEBUG

The DEBUG statement had some changes in z/OS V1R2. Now it can be defined in TELNETPARMS, TELNETGLOBALS or PARMSGROUP block. The parameters that can be used are:

▶ OFF

When OFF is specified, no debug records are issued except for connection drops due to timeouts or errors.

▶ SUMMARY

When SUMMARY is specified, a summary debug message (EZZ6034I) is issued indicating major state changes. It provides tracking of connection status. A summary message is written when:

- A connection request is accepted by Telnet.
- Connection negotiation is complete.
- A session is established with the host application.
- A session is dropped.
- A connection is dropped.

▶ DETAIL

When DETAIL is specified, a detail debug message (EZZ6035I) is written whenever a reportable error is detected in Telnet. Summary messages are also written when DETAIL is specified. The DEBUG DETAIL statement may be needed if the DEBUG SUMMARY messages do not provide enough information to solve a problem. In addition to the summary messages listed above, DEBUG DETAIL will issue a message at the time of failure that displays the client IP address and port, connection ID, Telnet LU name, detecting module name, unique return code and a brief explanation, and additional parameters if relevant.

▶ TRACE

When TRACE is specified, data to and from the client and to and from VTAM is displayed by the debug message EZZ6035I. Detail and summary messages are also written when TRACE is specified. The TRACE option allows you to quickly see why a client is not connecting or why a session hangs. Once TRACE is added via OBEYFILE, the first client assigned tracing will be the only client traced. Another connection cannot be traced until the client currently being traced is dropped.

▶ JOBLLOG

When JOBLLOG is specified, the debug messages are routed to the joblog instead of the console.

▶ CONSOLE

When CONSOLE is specified, the debug messages are routed to the operator console and to the teleprocessing console in addition to being sent to the joblog.

The DEBUG parameter may cause flooding of the operator's console. Console flooding concerns can be dealt with in several ways.

- ▶ DEBUG messages are, by default, assigned to routing code 11 - the joblog. The DEBUG option JOBLLOG can be used for the same effect. However, the master console also receives routing code 11 messages by default. To stop the messages from going to the

master console, issue **VARY CN(01),DROUT=(11)**, which drops routing code 11 from the console. The other DEBUG option, **CONSOLE**, will direct the messages to the master console, routing code 2, and the teleprocessing console, routing code 8.

- ▶ If DEBUG messages are being used primarily for problem diagnosis, the **OBEYFILE** command can be used to keep the number of messages low. Bring up Telnet initially without DEBUG coded. When a problem appears, issue an **OBEYFILE** for a Telnet profile that includes the DEBUG statement. Only new connections to the new profile will produce messages. Once data is obtained, issue another **OBEYFILE** for a Telnet profile that omits the DEBUG statement.
- ▶ If the client identifier of the client having the problem is known, include DEBUG in a **PARMSGROUP** statement. Using **PARMSMAP**, map that group to the client.

The **VARY TCPIP,,TELNET,DEBUG,OFF** command can be issued to turn off DEBUG for all connections associated with all profiles, including the current profile.

8.4.2 Abend Trap

The Abend Trap feature allows you to set up for a dump of the TCP address space at the time of failure in Telnet. This is a new feature of z/OS V1R2 CS IP. This command provides abend dumps based on a return code being set in a given module.

The **VARY TCPIP,,TELNET,ABENDTRAP,module,rcode,instance** command can be used to set up an abend based on the variables specified. **ABENDTRAP** has three variables:

- ▶ module
- ▶ rcode
- ▶ instance

Below is a sample of the **ABENDTRAP** command:

```
V TCPIP,TCPCS6,T,ABENDTRAP,EZBTTRCV,1001
EZZ0060I PROCESSING COMMAND:VARY TCPIP,TCPCS6,T,ABENDTRAP,EZBTTRCV,1001
EZZ6013I TELNET COMMAND ABENDTRAP EZBTTRCV COMPLETE
```

Figure 8-33 *ABENDTRAP* command

Note: The Abend Trap deactivates after its first occurrence.

8.4.3 CTRACE

CTRACE, with only the Telnet option, gives very complete information about the Telnet processes. To debug almost any Telnet problem, no other **CTRACE** option is needed. Generally, the other options simply take up space creating a trace-wrap condition more quickly. If the problem is data related, use the **FULLDATATRACE** statement to trace all the data coming into and leaving Telnet rather than tracing only the first 64 bytes of data. **FULLDATATRACE** will cause a trace-wrap condition more quickly so should be set only if needed. It should be set in **PARMSGROUP** instead of **TELNETPARMS** if a subset of clients can be identified. For transform problems, the **DBCSTRACE** statement in **TELNETPARMS** should be used to produce more trace entries in the **SYSPRINT** and **TNDBCSE** data sets.



Sample REXX to create HOSTS.LOCAL from /etc/hosts

This sample REXX program converts an /etc/hosts local file into a HOSTS.LOCAL file and executes the TCP/IP for MVS MAKESITE utility to build the corresponding datasetprefix.HOSTS.ADDRINFO and datasetprefix.HOSTS.SITEINFO data sets.

```
/* REXX */
call syscalls 'ON'          /* Allow OMVS REXX services          */

hostfile = '/etc/hosts'    /* Input hosts file                */
tcpiphlq = 'TCPIP.OMVS'   /* xxxxINFO data sets HLQ         */
dotrace = 0                /* Set to 1 for application trace hooks */

/* Let's start by reading the /etc/hosts file into a REXX stem */

address syscall "readfile" hostfile "hostline."
if retval = -1 then do
  say hostfile 'not read, errno='errno' - errnojr='errnojr
  exit(8)
end
if hostline.0 = 0 then do
  say hostfile 'is empty - no lines read'
  exit(8)
end

/* Create or allocate existing &userid.HOSTS.LOCAL data set */

If sysdsn(hosts.local) ~= 'OK' then do
  address TSO "alloc fi(hostloc) da(hosts.local) lrecl(255),
  blksize(0) dsorg(ps) recfm(v b) new catalog"
  alcrc = rc
  If alcrc > 0 then do
    say 'Allocation of new 'userid().HOSTS.LOCAL failed - rc = 'alcrc
    exit(alcrc)
  end
end
else do
```

```

    address TSO "alloc fi(hostloc) da(hosts.local) shr"
end

/* Process /etc/hosts lines and create stem var for HOSTS.LOCAL */

oix = 0
do i=1 to hostline.0
  If dotrace then say 'Trace input ==> 'hostline.i
  if words(hostline.i) > 0 then do
    if left(word(hostline.i,1),1) = '#' then iterate
    if words(hostline.i) = 1 |,
    left(word(hostline.i,2),1) = '#' then do
      say 'The following line from 'hostfile' is not valid syntax:'
      say '==> 'hostline.i
      iterate
    end
    oix = oix+1
    outline.oix = 'HOST:|'|word(hostline.i,1)|'|':
    dropline = 0
    do x=2 to words(hostline.i)
      if left(word(hostline.i,x),1) = '#' then leave
      newdata = word(hostline.i,x)|'|',
      newlength = length(outline.oix) + length(newdata)
      if newlength+4 > 255 then do
        say 'The following output line is too long and will be dropped:'
        say '==> 'outline.oix
        dropline = 1
        leave
      end
      else do
        outline.oix = outline.oix|'|word(hostline.i,x)|'|',
      end
    end
    if dropline then
      oix = oix-1
    else do
      outline.oix=left(outline.oix,length(outline.oix)-1)
      outline.oix = outline.oix|'|:':
      If dotrace then say 'Trace output ==> 'outline.oix
    end /*Test for dropping line */
  end /* Input-line with data */
end /* Do-loop for input lines */
outline.0 = oix

/* Write lines to &userid.HOSTS.LOCAL and run MAKESITE */

"EXECIO * DISKW HOSTLOC (STEM OUTLINE. FINIS"
address TSO "MAKESITE HLQ="tcpiph1q

exit(0)

```

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 on ordering these publications, see “How to get IBM Redbooks” on page 224.

- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- ▶ *OS/390 eNetwork Communications Server for V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing*, SG24-6516 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date July 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date July 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002))
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840 (redpiece available at <http://www.ibm.com/redbooks> (expected redbook publish date August 2002))
- ▶ *TCP/IP in a Sysplex*, SG24-5235
- ▶ *Managing OS/390 TCP/IP with SNMP*, SG24-5866
- ▶ *Secure e-business in TCP/IP Networks on OS/390 and z/OS*, SG24-5383
- ▶ *TCP/IP Tutorial and Technical Overview*, GG24-3376
- ▶ *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- ▶ *Networking with z/OS and Cisco Routers: An Interoperability Guide*, SG24-6297

Other resources

These publications are also relevant as further information sources:

- ▶ *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800
- ▶ *z/OS V1R2.0 UNIX System Services User's Guide*, GA22-7801
- ▶ *z/OS V1R1.0-V1R2.0 MVS Initialization and Tuning Guide*, SA22-7591
- ▶ *z/OS V1R2.0 C/C++ Programming Guide*, SC09-4765
- ▶ *z/OS V1R2.0 C/C++ Run-Time Library Reference*, SA22-7821
- ▶ *z/OS V1R2.0 CS: IP Migration*, GC31-8773
- ▶ *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775

- ▶ *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776
- ▶ *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780
- ▶ *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781
- ▶ *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 1 (EZA)*, SC31-8783
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 2 (EZB)*, SC31-8784
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 3 (EZY)*, SC31-8785
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 4 (EZZ-SNM)*, SC31-8786
- ▶ *z/OS V1R2.0 CS: IP Application Programming Interface Guide*, SC31-8788

Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ The z/OS Web pages
<http://www-1.ibm.com/servers/eserver/zseries/zos/installation/installz12.html>
- ▶ z/OS V1R2 Installation Planning Wizard
<http://www-1.ibm.com/servers/eserver/zseries/zos/wizards/ipw/ipwv1r2/>
- ▶ z/OS IP Configuration Wizard
<http://www.ibm.com/eserver/zseries/zos/wizards>
- ▶ SSL protocol pages
<http://home.netscape.com/eng/ss13/ssl-toc.html>
- ▶ Encryption methodology pages
<http://www.verisign.com/repository/crptintr.html>
<http://www.verisign.com/client/about/introCryp.html>

How to get IBM Redbooks

Search for additional Redbooks or Redpieces, view, download, or order hardcopy from the Redbooks Web site:

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become Redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

Symbols

/etc/hosts 106, 109
/etc/inetd.conf 112
/etc/networks 108
/etc/protocol 111
/etc/resolv.conf 98
/etc/services 111
@@@@@IPADDR 208
@@@@@DATE 208
@@@@@RUNAME 208
@@@@@TIME 208
@@@@@SENSE 208
@@@LUNAME 208
@@@PRT 208

Numerics

127.0.0.1 69, 87
14.0.0.0 69, 87
3270 data stream 143, 145
3270 DBCS transform mode 180

A

addressing family 24
AF_INET 3, 22, 23, 24
AF_INET socket 6, 23, 24
AF_INET socket address 24
AF_INET transport provider 25
AF_UNIX 22, 24, 25
AF_UNIX socket 22
ALGPRINT 66
alias for host name 109
ALLOWAPPL 190, 203
alternate port number 112
anchor configuration data set 98, 123
AnyNet Sockets over SNA 3
APF authorization 53
API interface 18
APIs 13
APPC/MVS 18
architecture, OS/390 IP 8
ASSORTEDPARMS 74

B

BEGINVTAM 188
BPX.SUPERUSER 43
BPXAS 33
BPXOINIT 33
BPXPARM, CINET 28
BPXPARM, INET 26, 27
BPXPRMxx 17, 27, 28, 29, 30, 65
BPXPRMxx, CINET 26
BPXTCINT 28

BPXTIINT 30, 65
BPXUIINT 30
branding 2, 18
BSD format 107
BUFFERPOOL statements 90

C

canonical mode 22
CEEDUMP 66
Channel Data Link Control (CDLC)
 description 10
Channel Protocols 10
Channel-to-Channel (CTC) 10
character string substitution 208
chcp command 139, 140
CICS 59
CINET 25, 26, 28
 and BPRPRMxx 30
 PFS 120
client IP address 208
client port number 208
Common INET Physical File System (CINET) 26, 120
Common Link Access to Workstation (CLAW) 10
Common Storage 90
Communication Storage Manager (CSM) 11
Comp IDs 53
Component IDs 53
component trace 158
configuration data set names 59
console commands 180
converged sockets 28
CONVXLAT 142
cookbook for creating multiple stacks 129
cron 19
Cross-System Coupling Facility (XCF) 12
cryptography 179
CSA 64, 90
C-sockets 13
CTIEZB00 114, 158
CTRACE 14, 158, 165
CTRLCONN 139
current date 208
current time 208
customization 49

D

Daemons 19
data trace 163, 165
DATASETPREFIX 59, 98, 110, 123
DATE 208
DATTRACE 163
DBCS 142, 178, 212
DBCS transform mode 180, 183
default directory path 46

- default group 56
- default user 21, 56
- default USS table 209
- DEFAULTAPPL 190, 202
- Direct Memory Access (DMA) 11, 12
- directory paths, EZAOEMDR 61
- DISCONNECTABLE 203
- DISPLAY 215
- display 180
- DISPLAY TCPIP command 149
- Domain Name System 107
- DYNAMIC XCF
 - IPCONFIG definition 89

E

- enablement, product 61
- Enterprise Extender 13
- ENTRYPOINT 27, 28, 65
- ENVAR 98, 104
- Ephemeral Ports 122
- explicit data set allocation 59
- EZACFSM1 77
- EZAGETIN 140
- EZAOEMDR 61
- EZAZSSI 63, 64
- EZAZSSI JCL procedure 65
- EZBPFINI 27, 30, 65
- EZBTJUST 203, 209
- EZZ4203I 65

F

- failing RU name 208
- File system attributes 40
- FM Header 182
- FMH 182
- FMIDs 53
- fork() 18
- forked address spaces 18
- FTP
 - security 58
- full-function mode, UNIX System Services 17, 71
- full-function, UNIX System Services 17
- full-screen support 178

G

- GENERIC 203
- generic resources 180
- Generic Server 121
- gethostbyname() 107
- getibmopt() 126
- getmain 64
- getservbyname() 112
- GID 20, 55
- Gigabit Ethernet 11
- graphical mode 22
- group ID 20, 55
- Groups 54

H

- HFS 18, 19
- Hierarchical File System 18, 19
- high level qualifier (HLQ) 59
- High Performance Data Transfer (HPDT) 11
- HiperSockets 12
- HNGROUP 190
- HOME Statement 87
- host name alias 109
- host name qualifier length 109
- host name qualifiers 109
- host name resolution 106
- hosts file
 - file format 107
 - file syntax 109
 - hosts file 106
 - name server 106
- HOSTS.ADDRINFO 110
- HOSTS.LOCAL 106, 109
- HOSTS.SITEINFO 110
- HSAS 51
 - need for HSAS 51

I

- ICHRIN03 56
- iconv functions 139
- iconv utility 145
- identity, MVS 20
- identity, UNIX 20
- IDTYPE=IPADDR 216
- IEASYSxx 29, 64
- IFA104I 61
- IFAPRDxx 61
- IKJTSOxx 64
- implicit data set allocation 59
- IMS 59
- INADDRANYPORT 65
- include files 81
- INCLUDE statement 92
- INET 25, 26
- inetd 19
- initapi() 126
- installation 49
 - BPXPRMxx 60
 - checklist 67
 - DATASETPREFIX 59
 - DFSMS 60
 - explicit data set allocation 59
 - high level qualifier (HLQ) 59
 - implicit data set allocation 59
 - LNKLST 63
 - LPALST 63
 - node name 63
 - PARMLIB 60, 62
 - planning the installation and migration 50
 - preinstallation 51
 - ripple 60
 - SCHEDxx 64
 - SMP/E 60

- SMS 60
 - steps 60
 - wave 60
- Integrated Sockets 27
- Integrated Sockets Physical File System (INET) 26
- INTERPTCP 190
- IP Assist 12
- IP names 1
- IPA 12
- IPADDR 208
- IPCS 165
- ISHELL 3
- ishell 37
- ISTEXCGR 181
- IUTSAMEH 12
- IVTPRMxx 65

L

- LAN Channel Station (LCS) protocol 11
- line mode 22, 143, 144
- LINEMODEAPPL 190, 202
- LNKAUTH 53
- LNKLST 63
- local hosts file 106
- local socket 25
- Logical Partitions (LPARs) 12
- login name 20
- LOGMODE 184
- LOGON command 179
- LOOPBACK 87
- loopback 69
- LPALST 63
- LUMAP 190, 193
- LUNAME 208

M

- MAKESITE 107, 109, 110
- mapping LUs 193
- MCS 180
- MD5 179
- message types 66, 172
- messages 66, 172
- MODIFY 212
- mount table 40
- Mounted file system 20
- MPC+ 11
- MSG07 190, 209
- msys 15
- msys for Setup 73
- MULTIPATH 89
- multiple AF_INET transport providers 25
- multiple ports support 179, 187
- Multiple Stacks 51, 117
 - _BPXK_SETIBMOPT_TRANSPORT environment variable 124
 - _iptcpn() 121
 - AF-INET PFS 120
 - back-end stack 118
 - capacity 120

- CINET PFS 120
- Common Internet Physical File System 120
 - cookbook 129
 - Ephemeral ports 122
 - example of two-stack configuration 130
 - fault-tolerant network attachments 119
 - front-end stack 118
 - Generic Server 121
 - INADDRANYCOUNT 122
 - INADDRANYPORT 122
 - inetd 121
 - network attachments 118
 - NETWORK DOMAINNAME parameter 121
 - performance 120
 - Physical File System (PFS) 120
 - PORTRANGE 122
 - reasons for multiple stacks 118
 - resolver configuration data sets 127
 - selecting a stack 123
 - setibmopt() socket call 121
 - SMF accounting 122
 - Sysplex 119
 - TSO users 126
 - VIPA 119
- multiple TCP/IP stacks 28
- MVPTSSI 63
- MVS identity 20

N

- name resolution 106
- name server and hosts file 106
- naming conventions, historical 1
- national language support 139
- NETSTAT 150
- Network Virtual Terminal 176
- NLS 139
 - 3270 data stream 143
 - 3270 full-screen mode 143
 - 3270 mode 143
 - chcp command 139, 140
 - code page 139
 - CONVXLAT 142
 - CTRLCONN 139
 - DBCS 142
 - EZAGETIN 140
 - iconv functions 139
 - iconv utility 145
 - line mode 143, 144
 - programmed symbols (PS) 143
 - SBDATACONN 139
 - Telnet 143
 - Telnet 3270 support 144
 - TN3270 144
 - translate table 140
 - XLATE 139
- non-canonical mode 22
- NVT 176

O

- OBEYFILE 208
- OBEYFILE and security 58
- OBEYFILE command 58, 92
- OE transport providers 23
- OESTACK 30
- offload 12
- OMPROUTE 13, 164
- OMVS 33
 - omvs display 33
- OMVS segment 55
- onetstat 150
- Open Shortest Path First 13
- OpenEdition Assembler Callable Services 13
- OPERCMDs, generic class 58
- oping 157
- OS/390 IP
 - introduction 1
 - names 1
- OS/390 IP APIs 13
- OS/390 IP, architecture 8
- OS/390 IP, overview 9
- OS/390 UNIX System Services 6
- OSA-Express 11
- OSPF 13
- otracert 157
- overview, OS/390 IP 9

P

- packet trace 163, 165
- PARMLIB 62
- Path Maximum Transmission Unit
 - IPCONFIG definition 89
 - RFC 1191 89
- pathname 25
- permission bits 20, 45
- PFS 24, 25, 120
- Physical File System (PFS) 24, 25, 120
- PING 157
- PKTRACE 163
- platform (VMCF/IUCV path) 3
- PORT 65, 190
- port number assignment 112
- Port Sharing 86
- PORT Statement 84
- POSIX 2
- preinstallation 49
- PRINT-EOJ 183
- printer support 204
- process 19
- PROCLIB 65
- product customization 49
- product enablement 61
- product registration 61
- profile replacement 181
- PROFILE.TCPIP 74
- PROGnn 63
- Program Directory 52, 59, 67
- programmed symbols (PS) 143, 144

- PROGxx in SYSx.PARMLIB 53
- protocol stack 24, 25
- PRT 208
- PRTGROUP 190
- PRTMAP 190
- PS 143, 144
- PSP bucket 52, 53
- PTP Samehost 12

Q

- QSESSION 203
- Queued Direct I/O 11
- Queued Direct I/O (QDIO) 11

R

- RACF 53, 54
 - RACF facility classes 54
 - RACF profiles 54
 - RACF resources 54
 - RACF STARTED class 56
 - RACF TERMINAL resource class 57
- raw mode 22
- Redbooks Web site 224
 - Contact us xiii
- registration, IFAPRDxx 61
- registration, product 61
- resolv.conf 98
- RESOLVE_VIA_LOOKUP compile symbol 106
- Resolver
 - RESOLVER_CONFIG environment variable 104
- Resolver configuration data sets 98, 127
- RESOLVER_CONFIG 184
- RESOLVER_CONFIG environment variable 104
- resource profiles 20
- restartable platform 63
- RESTRICTAPPL 190, 203
- RFC 1058 13
- RFC 1583 13
- RFC 1646 177
- RFC 1647 8, 177, 178
- RFC 1723 13
- RFC 952 format 107, 109
- RIP 13
- Root file system 20
- RS/6000 Communications Server 21
- RUNAME 208

S

- S806, abend code 65
- SBDATACONN 139
- SCHEDxx 64
- Secure Sockets Layer 179
- security 53
 - CICS 59
 - client 58
 - FTP 57, 58
 - IMS 59
 - NFS 57

- server 57
- SMTP 58
- SNMP 58
- security, NCS 57
- segment
 - OMVS 55
- SENSE 208
- sense code 208
- server (SLU) name 208
- service name 112
- setibmopt() 126
- SEZAINST 209
- SEZALINK 209
- SHA 179
- shared hosts file source 109
- shell 18
- shell access 21
- shell interface 18
- shell, ISHELL 18
- shell, OMVS 18
- single AF_INET transport provider 25
- SMF 180
- SMF record subtype assignment 122
- SMF records 16
- SMP/E requirements 59
- SMS 33
- sms display 33
- SMTP, security 58
- SNMP, security 58
- SNMPv1 58
- SNMPv2C 58
- SNMPv2u 58
- socket address 22, 24, 25
- Socket Addressing Families 22
- SOURCEVIPA 88
- spawn() 18
- spawned address spaces 18
- SPECIFIC 203
- SQA 64
- SSL 178, 179
- stack address space 3
- stand-alone PORT 186
- STARTED class in RACF 56
- STDENV 104
- SUBFILESYSTYPE 28
- superuser 21, 42, 56, 58
- symbolic links 46
- SYMDEF 78
- SYSCLONE system variable
 - definition 78
- SYSDEF 78
- SYSTCPD 183
- SYSTCPD DD name 98
- System Symbolics
 - case sensitivity 81, 83
 - coding 81
 - CONVSYM JCL utility 77
 - definition 78
 - INCLUDE file processing 77
 - OBEYFILE processing 77

- TCPIP.DATA processing 77
- TCPIP.PROFILE processing 77

T

- TCP/IP Base Functions
 - HOSTS.LOCAL 106
 - PROFILE.TCPIP 74
 - TCPIP.DATA 98
- TCP/IP data set names 59
- TCP/IP protocol stack 24
- TCP/IP system address space 3
- TCPCONFIG 84
- TCPIP.DATA 98, 123
- TCPIPJOBNAME 98, 123
- Telnet 143
 - BEGINVTAM 188
 - console commands 180
 - cryptography 179
 - DBCS 178
 - DBCS transform mode 180
 - FM Header 182
 - FMH 182
 - full-screen support 178
 - LOGMODE 184
 - LOGON command 179
 - LUMAP 193
 - MCS 180
 - multiple ports support 179
 - Network Virtual Terminal 176
 - NVT 176
 - PRINT-EOJ 183
 - profile replacement 181
 - RFC 1646 177
 - RFC 1647 177, 178
 - Secure Sockets Layer 179
 - SMF 180
 - SSL 178, 179
 - stand-alone PORT 186
 - Telnet 3270 176
 - TELNETPARMS 187
 - TN3270 176, 178
 - TN3270E 177, 178, 179
 - USS Messages 207
 - VT100 178
 - WLM 181
- telnet 175, 176
- telnet 3270 176
- telnet extensions 212
- telnetDEVICE 190
- TELNETPARMS 187
- terminal RACF class 57
- Thread 19
- TIME 208
- TN3270 176, 178
- TN3270E 8, 177, 178, 179
- TNDBCSCN 183
- TNDBCSE 183
- TNDBCSSL 183
- TNF 63
- TRACERTE 157

translate table 140
translation 140, 212
transport providers 3, 23, 25

Z
z/OS Managed System Infrastructure 15

U

UDPCONFIG 84
UID 20, 55
UNIX domain socket 25
UNIX identity 20
UNIX permission bits 45
UNIX shell 3
UNIX System Services
 full-function mode 17, 71
 history 2, 18
 minimum mode 17
UNIX, branding 18
Unknown RefID_idtel
 NLS 143
Unknown RefID_mapping
 LUs 193
user ID 20, 54, 55
user name 20
USS 207
USS sample table coding 209
USSTCP 190

V

VARY 212
vary 180
VARY ACT 212, 213
VARY INACT 212, 213
VARY OBEYFILE 212, 213
VARY QUIESCE 212, 214
VARY RESUME 212, 214
VARY STOP 212, 214
VARY TCPIP command 92
Virtual IP Address 88
Virtual IP Addressing
 IPCONFIG definition 88
VMCF 3, 63
VT100 178
VTAMLST 184

W

Web server 19
WLM 181, 213
WLMCLUSTERNAME 213
Workload Manager 18, 86, 181, 185
Workstation Operating Mode 22

X

X/Open 18
X/Open Portability Guides (XPG) 2
X_ADDR environment variable 110
X_SITE environment variable 110
XLATE 139
XPG 18
XPG4 2



Redbooks

Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration

(0.5" spine)

0.475" x 0.873"

250 x 459 pages



Communications Server for z/OS V1R2 TCP/IP Implementation Guide

Volume 1: Base and TN3270 Configuration



Redbooks

Understand and implement TCP/IP strategies related to CS for z/OS

Build and maintain your z/OS TCP/IP environment efficiently

Includes installation, base configuration, and TN3270

The Internet and enterprise-based networks have led to the rapidly increasing reliance upon TCP/IP implementations. The z/Series platform provides an environment upon which critical business applications flourish. The demands placed on these systems are ever-increasing and such demands require a solid, scalable, highly available, and highly performing operating system and TCP/IP component. z/OS and Communications Server for z/OS provide for such a requirement with a TCP/IP stack that is robust and rich in functionality. The *Communications Server for z/OS TCP/IP Implementation Guide* series provides a comprehensive, in-depth survey of CS for z/OS.

In Volume 1, we begin by providing an introduction to CS for z/OS. We include a survey on the evolution of what was once known as TCP/IP for MVS. We cover issues involved in using UNIX System Services as well as installation and base configuration of CS for z/OS. We further discuss other stack-related issues such as language support and multi-stack environments. Finally, because the TN3270 Server is so closely integrated with the stack, this volume details the intricacies of the server.

Because of the broad scope of CS for z/OS, this volume is not intended to cover all aspects of it. The main goal is to provide sufficient detail to install and initialize the TCP/IP stack. Additionally, this volume covers all stack-related issues. That is, anything that is system- or stack-related falls into the realm of this volume. For more advanced information, including routing and network interfaces, please refer to the other volumes in the series.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-5227-03

ISBN 0738424153