IBM

Communications Server for z/OS V1R9 TCP/IP Implementation Volume 2: Standard Applications

- Learn about CS for z/OS TCP/IP standard applications
- TCP/IP standard application implementation examples provided
- Leverage TCP/IP standard applications for your needs

Bill White
Gwen Dente
Octavio L Ferreira
Shizuka Katoh
Shuo Ni
Yohko Ojima
Joel Porterie
Larry Templeton

Redbooks

ibm.com/redbooks

Archived

Archived

International Technical Support Organization

CS for z/OS V1R9 TCP/IP Implementation Volume 2: Standard Applications

February 2008

Archived

SG24-7533-00

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

Archived

First Edition (February 2008)

This edition applies to Version 1, Release 9 of Communications Server for z/OS.

© Copyright International Business Machines Corporation 2008. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices ix
Trademarks x

Preface xi
The team that wrote this book xii
Become a published author xiii
Comments welcome xiii

Chapter 1. Syslogd 1
1.1 Conceptual overview of syslogd 2
1.1.1 What is syslogd? 2
1.1.2 How does syslogd work? 3
1.1.3 How can syslogd be applied? 3
1.2 Log messages to different files and to a single file 4
1.2.1 Description of logging to multiple files and to a single file 4
1.2.2 Configuration of multiple files and a single file 5
1.2.3 Verification of multiple files and a single file 8
1.3 Starting two syslogd instances 9
1.3.1 Description of two syslogd instances 9
1.3.2 Configuration of two syslogd instances 9
1.3.3 Verification for running two syslogd instances 11
1.4 Problem determination for syslogd logging 12
1.5 Additional information sources for syslogd 13

Chapter 2. TN3270E Telnet server 15
2.1 Conceptual overview of the TN3270E server 16
2.1.1 What is the TN3270E server 16
2.1.2 How does the TN3270E server work 17
2.1.3 How can the TN3270E server be used 21
2.2 TN3270E server in a single image 22
2.2.1 Description of our TN3270E server scenario 22
2.2.2 Configuration of the TN3270E server 23
2.2.3 Activation of the TN3270E server 31
2.2.4 Verification of the TN3270E server 32
2.2.5 Administration and management of the TN3270E server 43
2.3 Multiple TN3270E servers in a multiple image environment 52
2.3.1 Description of multiple TN3270E servers within the sysplex 53
2.3.2 Configuration of multiple TN3270E servers within the sysplex 55
2.3.3 Activation and verification of multiple TN3270E servers in the sysplex 61
2.4 Problem determination for the TN3270E servers 70
2.4.1 Reviewing the definition statements within the profile 70
2.4.2 Using TCP/IP and Telnet commands 70
2.4.3 Using the MSG07 statement in the TN3270 profile 75
2.4.4 Using SMF records to capture TN3270 connection activity 75
2.4.5 Using trace data 76
2.4.6 Tips for multiple TN3270E servers in a Parallel Sysplex environment 77
2.5 Additional information sources for the TN3270E server 77

Chapter 3. File Transfer Protocol 79

© Copyright IBM Corp. 2008. All rights reserved.

3.1 Conceptual overview of FTP . . . 80
3.1.1 What is FTP? . . . 80
3.1.2 How does FTP work? . . . 81
3.1.3 How can FTP be used? . . . 82
3.2 Basic FTP without security . . . 84
3.2.1 Description of basic FTP without security . . . 84
3.2.2 Planning for the basic FTP environment without security . . . 84
3.2.3 Configuration of basic FTP without security . . . 97
3.2.4 Activation and verification for basic FTP without security . . . 101
3.3 Multiple FTP servers in a sysplex . . . 108
3.3.1 Description of multiple FTP servers in a sysplex . . . 108
3.3.2 Configuration for multiple FTP servers in the sysplex . . . 110
3.3.3 Activation and verification of FTP servers within sysplex . . . 113
3.4 FTP client via batch . . . 119
3.4.1 Description of FTP client via batch . . . 120
3.4.2 Configuration of FTP client via batch . . . 120
3.4.3 Activation and verification of FTP client batch job . . . 121
3.5 Miscellaneous configurations of FTP . . . 122
3.5.1 FTP client via the FTP client API - REXX support . . . 122
3.5.2 A single generic FTP server in a multiple stack z/OS image . . . 123
3.5.3 FTP network management interface with SMF . . . 123
3.6 Problem determination for FTP . . . 124
3.7 Additional information sources for FTP . . . 124

Chapter 4. Simple Network Management Protocol . . . 127
4.1 Conceptual overview of SNMP . . . 128
4.1.1 What is SNMP? . . . 128
4.1.2 How does SNMP work? . . . 129
4.1.3 How can SNMP be applied? . . . 130
4.2 z/OS SNMP agent . . . 131
4.2.1 Description of the z/OS SNMP agent . . . 132
4.2.2 Configuration of the z/OS SNMP agent . . . 133
4.2.3 Activation and verification of the z/OS SNMP agents . . . 141
4.3 z/OS SNMP subagents . . . 142
4.3.1 Description of SNMP subagents . . . 142
4.3.2 Configuration of SNMP subagents . . . 143
4.3.3 Activation and Verification of SNMP subagents . . . 145
4.4 z/OS SNMP client command . . . 148
4.4.1 Description of the SNMP client commands . . . 148
4.4.2 Configuration tasks for the SNMP client commands . . . 148
4.4.3 Using the osnmp/snmp z/OS UNIX command . . . 149
4.5 Problem determination for the SNMP facilities . . . 156
4.6 Additional information sources for SNMP . . . 157

Chapter 5. IP printing . . . 159
5.1 Conceptual overview of IP printing . . . 160
5.1.1 What is IP printing? . . . 160
5.1.2 How does IP Printing work? . . . 161
5.1.3 How can IP Printing be applied? . . . 161
5.2 LPR/LPD . . . 162
5.2.1 Description of LPR/LPD . . . 163
5.2.2 Configuration tasks for LPR/LPD . . . 163
5.2.3 Activation and verification of LPR/LPD . . . 165

5.3 Infoprint Server . . . 167
5.3.1 Description of the Infoprint Server. . . . 167
5.3.2 Configuration of Infoprint Server . . . 170
5.4 Problem determination for LPR/LPD. . . . 176
5.5 Additional information sources for IP printing . . . 183

Chapter 6. INETD . . . 185
6.1 Conceptual overview of INETD . . . 186
6.1.1 What is INETD? . . . 186
6.1.2 How does INETD work? . . . 187
6.1.3 How can INETD be applied?. . . . 187
6.2 A single INETD setup . . . 188
6.2.1 Description of the INETD setup. . . . 188
6.2.2 Configuration tasks for INETD setup. . . . 188
6.2.3 Activation and verification of INETD . . . 190
6.3 Problem determination for INETD . . . 194
6.4 Additional information sources for INETD . . . 195

Chapter 7. z/OS mail servers . . . 197
7.1 Conceptual overview of z/OS mail applications . . . 198
7.1.1 What are z/OS mail services?. . . . 198
7.1.2 How do z/OS mail services work?. . . . 199
7.1.3 How can z/OS mail services be applied? . . . 199
7.2 z/OS SMTP as a mail server. . . . 200
7.2.1 Description of z/OS SMTP server . . . 200
7.2.2 Configuration tasks for the z/OS SMTP server . . . 203
7.2.3 Verification of the z/OS SMTP server . . . 212
7.3 Sendmail and popper as mail servers. . . . 213
7.3.1 Description of sendmail and popper . . . 213
7.3.2 Configuration tasks for sendmail and popper . . . 217
7.3.3 Verification of sendmail and popper setup . . . 221
7.4 Using sendmail as a client . . . 227
7.4.1 Description of the sendmail client . . . 227
7.4.2 Configuration tasks for the sendmail client . . . 227
7.4.3 Verification of the sendmail client . . . 230
7.5 Problem determination for the mail facilities . . . 231
7.5.1 Problem determination tasks for the z/OS SMTP server . . . 231
7.5.2 Problem determination for sendmail and popper . . . 232
7.5.3 Problem determination for the sendmail client . . . 233
7.6 Additional information sources for mail servers. . . . 233

Chapter 8. z/OS UNIX Telnet server . . . 235
8.1 Conceptual overview of otelnetd . . . 236
8.1.1 What is otelnetd? . . . 236
8.1.2 How does otelnetd work? . . . 236
8.1.3 How can otelnetd be applied?. . . . 237
8.2 z/OS UNIX Telnet server implementation . . . 237
8.2.1 Description of the otelnetd server . . . 238
8.2.2 Configuration tasks for otelnetd . . . 238
8.2.3 Activation and verification of otelnetd . . . 242
8.3 Problem determination for otelnetd . . . 243
8.4 Additional information sources for otelnetd . . . 243

Chapter 9. Remote execution . . . 245

9.1 Conceptual overview of remote execution 246
9.1.1 What is remote execution? 246
9.1.2 How does remote execution work? 247
9.1.3 How can remote execution be applied? 249
9.2 TSO remote execution server 250
9.2.1 Description of TSO remote execution server 251
9.2.2 Configuration tasks for TSO remote execution server 252
9.2.3 Activation and Verification of TSO remote execution server 257
9.3 z/OS UNIX remote execution server 257
9.3.1 Description of z/OS UNIX remote execution server 258
9.3.2 Configuration tasks for z/OS UNIX remote execution server 258
9.3.3 Activation and verification of z/OS UNIX remote execution server 260
9.4 REXEC TSO client command using user ID/password 262
9.4.1 Description of REXEC TSO with user ID and password 262
9.4.2 Configuration of REXEC TSO with user ID and password 262
9.4.3 Verification of REXEC TSO with user ID and password 264
9.5 REXEC TSO client command using the NETRC data set 266
9.5.1 Description of REXEC TSO client using NETRC 266
9.5.2 Configuration of REXEC TSO client using NETRC 266
9.5.3 Verification of REXEC TSO client using NETRC 268
9.6 REXEC UNIX client command 271
9.6.1 Description of the REXEC UNIX client command 271
9.6.2 Configuration of the REXEC UNIX client command 272
9.6.3 Verification of the REXEC UNIX client command 273
9.7 Problem determination for z/OS remote execution facilities 274
9.7.1 Problem determination for TSO remote execution 274
9.7.2 Problem determination for REXEC TSO with user ID and password 275
9.7.3 Problem determination of REXEC TSO using NETRC 275
9.7.4 Problem determination for the REXEC UNIX client command 276
9.8 Additional information sources on remote execution and remote shell 277

Chapter 10. Domain Name Service 279
10.1 Conceptual overview of the DNS name server 280
10.1.1 What is Domain Name Service? 280
10.1.2 How does Domain Name Service work? 281
10.1.3 How can Domain Name Service be applied? 282
10.2 Authoritative DNS server 283
10.2.1 Description of an authoritative DNS server 283
10.3 Caching-only DNS server 284
10.3.1 Description of a caching-only DNS server 285
10.3.2 Configuration of a caching-only DNS server 285
10.3.3 Activation and verification of a caching-only DNS server 290
10.4 Automated domain name registration 294
10.4.1 Description of ADNR 295
10.4.2 Configuration of ADNR 296
10.4.3 Activation and verification of ADNR 300
10.5 Problem determination for DNS service 304
10.5.1 Problem determination for a caching-only DNS server 304
10.5.2 Problem determination for ADNR 306
10.6 Additional information sources for DNS 309

Appendix A. Environment variables 311
Description of the environment variable information 312

Native MVS API environment 312
z/OS UNIX API environment 313
z/OS UNIX System Services environment variables 313
Language Environment, environment variables 314
Application-specific environment variables 315
Setting environment variables 321

Appendix B. Sample files provided with TCP/IP 323
Sample files by component 324

Appendix C. Configuration files: TN3270E standalone started task scenario 331
SC30 files for TN3270 standalone started task scenario 332
SC30 TN3270B Server PROC for TN3270 standalone started task 332
SC30 TN3270B Server PROFILE for TN3270 standalone started task 332
SC30 TCPIPB stack PROC for TN3270 standalone started task 334
SC30 TCPIPB stack PROFILE for TN3270 standalone started task 335
SC30 OMPROUTE PROC for TN3270 standalone started task 338
SC30 OMPROUTE STDENV file for TN3270 Standalone Task Scenario 338
SC30 OMPROUTE CONFIG for TN3270 standalone started task 338

Appendix D. Configuration files: Multiple TN3270E Telnet servers and SD scenario 343
SC30 files for multiple TN3270E Telnet servers and SD scenario 344
SC30 TN3270B Server PROC for multiple TN3270E Telnet servers and SD 344
SC30 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD 344
SC30 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD 348
SC30 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD 348
SC30 OMPROUTE PROC for multiple TN3270E Telnet servers and SD 353
SC30 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD 353
SC30 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD 353
SC31 files for multiple TN3270E Telnet servers and SD scenario 356
SC31 TN3270B Server PROC for multiple TN3270E Telnet servers and SD 356
SC31 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD 357
SC31 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD 361
SC31 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD 361
SC31 OMPROUTE PROC for multiple TN3270E Telnet servers and SD 366
SC31 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD 366
SC31 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD 367

Appendix E. FTP and translation tables 371
Conceptual overview of FTP translation 372
What is translation? 372
How does translation work? 372
How can FTP translation be applied? 375
Using the RFC2389 and RFC2640 FTP features 377
RFC 2389 - Feature negotiation 378
RFC2640 - FTP Internationalization 378
Requirements to implement these RFCs 378
Selecting translation tables 379
Using the QUOTE SITE subcommand 379
Using the TRACE option at the server 380
Using the DEBUG option at the client 382
Using the TRANSLATE sub command 382
Setting a DBCS transfer mode 383
Enabling Unicode transfer mode 385

. 385

Appendix F. Our implementation environment 387
The environment used for all four books 388
Our focus for this book 389

Related publications 391
IBM Redbooks 391
Other publications 391
Online resources 393
How to get IBM Redbooks 393
Help from IBM 393

Index 395

Archived

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.

© Copyright IBM Corp. 2008. All rights reserved.

Trademarks

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

AIX®
C/370™
CICS®
Domino®
DPI®
FICON®
HiperSockets™
IBM®
IMS™
IP PrintWay™
Language Environment®
Lotus Notes®
Lotus®
MVS™
NetSpool™
NetView®
Notes®
Parallel Sysplex®
Print Services Facility™
PrintWay™
RACF®
Redbooks®
Redbooks (logo) ®
Resource Link™
REXX™
System z™
System z9®
System/360™
Tivoli®
VTAM®
WebSphere®
z/OS®
z9™

The following terms are trademarks of other companies:

PostScript, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

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

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

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

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

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

Archived

Preface

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

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

The *Communications Server for z/OS TCP/IP Implementation* series of IBM Redbooks® publications provides easy-to-understand, step-by-step guidance about enabling the most commonly used and important functions of CS for z/OS TCP/IP.

In this book, *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 2: Standard Applications*, SG24-7533, we provide useful implementation scenarios and configuration recommendations for many of the TCP/IP standard applications supported by the z/OS Communications Server.

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

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535

The publications *z/OS Communications Server: IP Configuration Guide*, SC31-8775, *z/OS Communications Server: IP Configuration Reference*, SC31-8776, and *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, contain comprehensive descriptions of the individual parameters for setting up and using the functions described in this book. They also include step-by-step checklists and supporting examples.

It is not the intent of this book to duplicate the information in those manuals, but to complement them with practical implementation scenarios that can be useful in your environment. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771. For complete details, we encourage you to review the documents referred to in the additional resources section at the end of each chapter.

© Copyright IBM Corp. 2008. All rights reserved.

The team that wrote this book

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

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

Gwen Dente has spent 27 years with IBM, focusing on mainframe networking and security support for customers and the IBM technical community. She is a Consulting IT Specialist with IBM Advanced Technical Support at the Washington Systems Center in Gaithersburg, Maryland, USA. Gwen presents frequently at SHARE and other IBM technical conferences, and has shared authorship of multiple IBM Redbooks publications.

Octavio L Ferreira is a Senior IT Specialist in IBM Brazil. He has 28 years of experience in IBM software support. His areas of expertise include z/OS Communications Server, SNA and TCP/IP, and Communications Server on all platforms. For the last 10 years, Octavio has worked at the Area Program Support Group, providing guidance and support to clients and designing networking solutions such as SNA/TCP/IP Integration, z/OS Connectivity, Enterprise Extender design and implementation, and SNA-to-APPN migration. He has also co-authored other IBM Redbooks publications.

Shizuka Katoh is an Advisory IT Specialist working for Advanced Technical Support in IBM Japan. She has more than 10 years of experience the field of Mainframe Networking. Her areas of expertise include z/OS Communications Server, TCP/IP (IPv4 and IPv6), VTAM®, and OSA. Shizuka specializes in High Availability Data Center networking solutions with Parallel Sysplex® environments.

Shuo Ni is an Advisory IT Specialist in IBM Global Services, China. Her areas of expertise include z/OS, TCP/IP, VTAM (Subarea and APPN), OSA-Express, HiperSockets™, WLM, and CICS®. For the last six years, Shuo has provided onsite service support to mainframe clients in southern China for z/OS Communication Server (including Enterprise Extender and CICS TCP/IP Sockets), as well as other mainframe subsystems.

Yohko Ojima is an IT Specialist in IBM Japan with five years of experience in the area of Enterprise Networking. She specializes in TCP/IP and SNA networks with Communications Server products, routers, and switches. Yohko's responsibilities include providing technical support and reviewing the networking design for many customers, including Communications Server for z/OS (TCP/IP and VTAM), Communications Server for distributed servers, OSA, and Communication Controller for Linux® on System z.

Joel Porterie is a Senior IT Specialist who has been with IBM France for 30 years. He works for Network and Channel Connectivity Services in the EMEA Product Support Group. His areas of expertise include z/OS, TCP/IP, VTAM, OSA-Express, and Parallel Sysplex. Joel has taught OSA-Express and FICON® problem determination classes, and has provided onsite assistance in these areas in numerous countries. He has also co-authored many other IBM Redbooks publications.

Larry Templeton is a Network Architect with IBM Global Services, Network Outsourcing. He has 38 years of experience with IBM Mainframe and Networking systems, consulting with clients around the world. His current responsibilities include architecting secure mainframe IP connectivity solutions, designing inter-company Enterprise Extender configurations, and assisting clients with high-availability data center implementations. Larry has also co-authored other IBM Redbooks publications.

Thanks to the following people from the International Technical Support Organization, Poughkeepsie Center for their contributions to this project: David Bennin, Ella Buslovich, Rich Conway, and Bob Haimowitz.

As is always the case with any complex technical effort, success would not have been possible without the advice, support, and review of many outstanding technical professionals. We are especially grateful for the significant expertise and contributions of content to this book from the Communications Server for z/OS development team, especially Kristine Adamson, Doris Bunn, David Bruton, Alfred Christensen, Gus Kassimis, Mary Lisica, and Gary McAfee.

Finally, we would like to thank the authors of the previous *Communications Server (CS) for z/OS TCP/IP Implementation* series for creating the groundwork for this series; they include: Rama Ayyar, Valirio Braga, Gilson Cesar de Oliveira, Octavio Ferreira, Adi Horowitz, Michael Jensen, Sherwin Lake, Bob Louden, Garth Madella, Yukihiko Miyamoto, Roland Peschke, Joel Porterie, Marc Price, Larry Templeton, Rudi van Niekerk, and Thomas Wienert.

Become a published author

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

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

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

`ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us!

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

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

  `ibm.com/redbooks`

- Send your comments in an e-mail to:

  `redbooks@us.ibm.com`

- Mail your comments to:

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

Archived

Archived

1

Syslogd

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

The following syslogd topics are discussed in this chapter.

Section	Topic
1.1, “Conceptual overview of syslogd” on page 2	The basic concepts of syslogd.
1.2, “Log messages to different files and to a single file” on page 4	How to configure syslogd to log messages to different files, based on jobname, and also send all messages to a single file without regard to jobname.
1.3, “Starting two syslogd instances” on page 9	How to configure and run two instances of syslogd: one in local mode, the other in network mode.
1.4, “Problem determination for syslogd logging” on page 12	How to determine why certain log files might be empty.
1.5, “Additional information sources for syslogd” on page 13	References to additional syslogd documentation materials.

© Copyright IBM Corp. 2008. All rights reserved.

1.1 Conceptual overview of syslogd

Syslogd is one of the standard applications provided with the z/OS Communications Server. The relationship of syslogd to applications and to other hosts for which it provides logging services is illustrated in Figure 1-1. Syslogd running in local mode (**`syslogd -i`**) can collect messages from local applications and either record them into a local file for that application, or forward them to another remote host for logging.

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

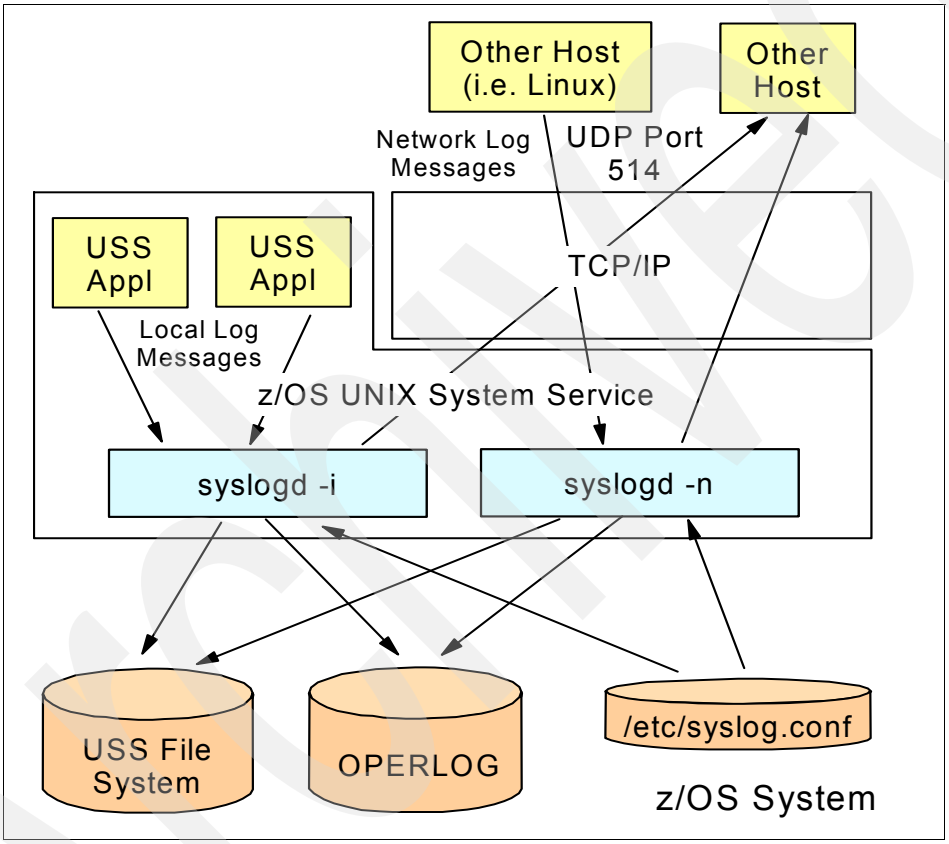


Figure 1-1 Syslogd relationship to applications for which it logs information

A single instance of syslogd can do both local and remote message processing if so desired.

1.1.1 What is syslogd?

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

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

1.1.2 How does syslogd work?

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

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

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

1.1.3 How can syslogd be applied?

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

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

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

Sending all messages to a single file

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

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

This configuration can be set up in a matter of minutes and is easy to understand. However, sending all messages to a single file makes it difficult to follow what is occurring in a single application over many hours.

Sending messages to different files based on job name

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

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

This configuration makes it very easy to find messages related to a particular application and follow the flow of events. This configuration also employs *syslogd isolation* by only allowing particular applications to write to syslogd facilities that should be reserved for system

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

Sending messages to different files and to a single file

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

Starting two instances of syslogd

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

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

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

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

1.2 Log messages to different files and to a single file

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

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

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

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

1.2.2 Configuration of multiple files and a single file

Perform these configuration tasks:

- Plan your configuration
- Create an /etc/syslog.conf configuration file
- Create a crontab entry for syslogd
- Start syslogd

Plan your configuration

Become familiar with and understand how to use the syslogd configuration parameters:

Syslogd facilities

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

Table 1-1 Facilities

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

Syslogd configuration file

Syslogd processing is controlled by a configuration file (usually named /etc/syslog.conf). A sample syslogd configuration file is included in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. The syslogd configuration file allows you to separate messages based on the user ID of the application generating the message, job name of the application generating the message, facility used by the application, and priority of the message generated by the application.

Archived

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

Syslogd parameters

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

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

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

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

  The start options -D and -F establish the *default* permissions for directories and files created *automatically*. The corresponding configuration options, -D and -F, specified in the configuration file, provide the ability to set the permission bits, on an individual rule basis, for directories (-D) and for files (-F). When specified, the configuration options override any default options established at startup.

  These two options, -D and -F, are specified in octal notation, and are valid only on syslog destinations that are regular UNIX file system files. These options require the use of the -c option as well. Not all permission bit combinations are valid. For complete details on restricted bit combinations, refer to the syslog daemon sections in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

- Local-only mode (-i option)

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

- Network-only mode (-n option)

  You can use the -n option as well. The -n option allows syslogd to receive messages over the network only.

- Disable host name resolution (-x option)

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

Syslogd isolation

Syslogd isolation refers to a configuration of syslogd that allows only certain job names to write log messages to particular syslogd destinations. The syslogd isolation feature is automatically enabled when job names are included as a filter option in the syslogd configuration file rules.

Create an /etc/syslog.conf configuration file

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

lines of the configuration file, starting from the line that reads: `THIS EXAMPLE STATEMENT IS UNCOMMENTED`, to the end of file. Then add the lines shown in Example 1-1.

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

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

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

Create a crontab entry for syslogd

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

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

Example 1-2 Crontab entries

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

Start syslogd

Add the lines in Example 1-3 to /etc/rc to start syslogd every time z/OS UNIX starts.

Example 1-3 /etc/rc entry to start syslogd

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

Alternatively, you can use the started task in Example 1-4 to start syslogd.

Example 1-4 Started task for SYSLOGD

```
//SYSLOGD PROC
//SYSLOGD EXEC PGM=SYSLOGD,REGION=0K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)',
//       'ENVAR("TZ=EST5EDT")/-c -i -D 0755 -F 0664')
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Archived

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

Note that RECFM=F must *not* be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in UNIX System Services. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

1.2.3 Verification of multiple files and a single file

To verify that syslogd is working correctly, go to the directory that contains the log files and check each file. Example 1-5 shows the output of **/bin/ls** that was issued while in our /var/log/2007/09/13 directory.

Example 1-5 Output of /bin/ls in our log directory

```
# cd /var/log/2007/09/20
# ls -l
total 48
-rw-------   1 IBMUSER  SYS1           0 Sep 20 21:04 ftp
-rw-------   1 IBMUSER  SYS1          86 Sep 20 21:12 inetd
-rw-------   1 IBMUSER  SYS1        9529 Sep 20 21:12 msgs
-rw-------   1 IBMUSER  SYS1           0 Sep 20 21:04 named
-rw-------   1 IBMUSER  SYS1        4468 Sep 20 21:10 omproute
-rw-------   1 IBMUSER  SYS1           0 Sep 20 21:09 sendmail
-rw-------   1 IBMUSER  SYS1           0 Sep 20 21:04 syslogd
```

Example 1-6 shows the contents of one of our log files.

Example 1-6 Contents of the omproute log file

```
Sep 20Sep 20 00:10:21 OMPROUTE omproute[83886100]: EZZ7800I OMPROUTE starting
Sep 20 00:10:22 OMPROUTE omproute[83886100]: EZZ7845I Established affinity with TCPIP
Sep 20 00:10:22 OMPROUTE omproute[83886100]: EZZ7817I Using default OSPF protocol 89
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7817I Using default OSPF protocol 89
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7838I Using configuration file:
/etc/omproute/omproute.conf
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7890I Non-broadcast ignored when
OSPF_Interface statement is a wildcard
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 10.1.1.20, name VIPA1L, index 0, flags 4041
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 10.1.2.20, name VIPA2L, index 0, flags 4041
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 10.1.2.21, name OSA2080L, index 1, flags 450
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 10.1.2.22, name OSA20A0L, index 2, flags 450
Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from stack,
address 10.1.3.21, name OSA20E0L, index 3, flags 450
Sep 20Sep 20 00:10:27 OMPROUTE omproute[83886100]: EZZ7883I Processing interface from
stack, address 10.1.3.22, name OSA20C0L, index 4, flags 450
```

1.3 Starting two syslogd instances

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

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

1.3.1 Description of two syslogd instances

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

However, because network syslogd messages are delivered via UDP, delivery of messages is not guaranteed. Moreover, messages may not be delivered to z/OS under some conditions. The use of IPSEC should be considered for protecting the syslog daemon's UDP port 514 when running in local or network-only mode. For more information, refer to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535.

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

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

1.3.2 Configuration of two syslogd instances

- Create /etc/syslog.conf configuration files
- Create crontab entries for syslogd
- Start syslogd

Create /etc/syslog.conf configuration files

We established only one configuration file to be shared by the two instances of syslogd. Example 1-7 shows our configuration file.

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

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

Example 1-7 /etc/syslog.conf configuration file for z/OS

```
##########################
#   For local-only mode   #
##########################
*.INETD*.*.* /var/log/%Y/%m/%d/inetd
*.FTP*.*.* /var/log/%Y/%m/%d/ftp
*.SYSLOGD*.*.* /var/log/%Y/%m/%d/syslogd -D 0755 -F 0664
*.OMP*.*.* /var/log/%Y/%m/%d/omproute
*.SENDMAIL.*.* /var/log/%Y/%m/%d/sendmail
*.NAMED*.*.* /var/log/%Y/%m/%d/named
# Add additional jobnames here as needed

##########################
#  For network-only mode  #
##########################
(wtsc30).daemon.debug /dev/operlog 1
# Add additional hostspecs here as needed

# Send a copy of all messages to the file named msgs_net
(wtsc30).*.* /var/log/%Y/%m/%d/msgs 2
```

In Example 1-7, note the following points:

- 1 Log messages from remote host wtsc30 can be written to /dev/operlog.
- 2 We defined remote host wtsc30 to prevent syslogd from receiving messages from unintended hosts.

As shown in Example 1-8, we added an entry 1 into the /etc/syslog.conf configuration file on our remote host, and then restarted the syslog daemon on it.

Example 1-8 /etc/syslog.conf entry on the remote host: wtsc30

```
*.*                                @wtsc31 1
```

You can specify the IP address instead of the host name.

Create crontab entries for syslogd

From the z/OS UNIX shell issue the command **`export EDITOR=oedit`**, and then issue the command **`crontab -e`**. Using the editor, add the two lines shown in Example 1-9. Save the file and exit the editor. These entries allow log files to be recreated daily.

Example 1-9 Crontab entries

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

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

Start syslogd

We issued z/OS UNIX shell commands as follows:

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

To start syslogd every time z/OS UNIX starts we added the same lines to /etc/rc as we issued, as shown in Example 1-10.

Example 1-10 /etc/rc entries to start syslogd

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

1.3.3 Verification for running two syslogd instances

To verify that the syslog daemons are working correctly, we can use a **ps -ef | grep syslogd** command and verify the log files in the directory match the defined configuration files. Example 1-11 shows a process list.

Example 1-11 Process list

```
BPXROOT @ SC31:/>ps -ef | grep syslogd
 BPXROOT     131319          1  - 20:38:05 ?         0:00 /usr/sbin/syslogd -f /
etc/syslog.conf -c -n -x 1
 BPXROOT   67240127          1  - 20:38:53 ?         0:00 /usr/sbin/syslogd -f /
etc/syslog.conf -c -i -D 0755 -F -664 2
```

In Example 1-11, note the following points:

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

Example 1-12 shows the output of an **ls** command that was issued while in the /var/log/2007/09/24 directory.

Example 1-12 Output of the ls command

```
BPXROOT @ SC31:/>cd /var/log/2007/09/20
BPXROOT @ SC31:/SC31/var/log/2007/09/20>ls -l
total 144
-rw-r--r--   1 BPXROOT  SYS1         857 Sep 20 20:41 ftp
-rw-r--r--   1 BPXROOT  SYS1           0 Sep 20 19:55 inetd
-rw-r--r--   1 BPXROOT  SYS1       46831 Sep 20 20:21 msgs
-rw-r--r--   1 BPXROOT  SYS1        6968 Sep 20 20:10 named
-rw-r--r--   1 BPXROOT  SYS1        6228 Sep 20 20:19 omproute
-rw-r--r--   1 BPXROOT  SYS1           0 Sep 20 19:55 sendmail
-rw-r--r--   1 BPXROOT  SYS1         706 Sep 20 20:38 syslogd
```

Example 1-13 shows the output of /dev/operlog on SC31 (also known as wtsc31). The log entries (`wtsc30.itso.ibm.com`) show that those messages were received from a remote host.

Example 1-13 /dev/operlog on SC31

```
/dev/operlog
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  INFO   :006: ...open_sock
et:  Try 1, connecting AF_INET socket number = 8, for address = 10.1.1.40, port
= 16310
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  INFO   :006: ...open_sock
et:  Issue select to test for complete or wait 1 seconds for timeout
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  INFO   :006: ...open_sock
et:  Exiting, socket number = 8
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  OBJERR :006: ...plfm_init
_SSL_conn:  unable to initialize SSL secure socket, rc = 420 (Socket closed by r
emote partner)
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  OBJERR :006: ...open_conn
ection:  Initialize SSL connection failed
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  INFO   :006: ...open_conn
```

```
ection:  Exiting
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  INFO   :006: ...APIInitia
lize:  Exiting
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  INFO   :006: ...APIInitia
lize:  Freeing Storage (146880F0)
Sep 21 15:39:58 wtsc30.itso.ibm.com Pagent?0397293?  OBJERR :006: ...papi_conn
ect:  APIInitialize failed
```

Example 1-14 shows the content of the syslogd log file.

Example 1-14 Syslogd log file

```
/var/log/2007/09/20/syslogd
BPXROOT @ SC31:/SC31/var/log/2007/09/20>cat syslogd
Sep 20 21:25:52 WTSC31 syslogd: FSUM1220 syslogd: restart
Sep 20 21:25:52 WTSC31 syslogd: FSUM1237 Job SYSLOGD1 running in local-only mode 1
Sep 20 21:25:52 WTSC31 syslogd: FSUM1232 syslogd: running non-swappable
Sep 20 21:26:04 WTSC31 syslogd: FSUM1220 syslogd: restart
Sep 20 21:26:04 WTSC31 syslogd: FSUM1238 Job SYSLOGD2 running in network-only mo 2
de
Sep 20 21:26:04 WTSC32 syslogd: FSUM1232 syslogd: running non-swappable
```

In Example 1-14, key log messages are as follows:

- 1 shows Job SYSLOGD1 running in local-only mode.
- 2 shows Job SYSLOGD2 running in network-only mode.

1.4 Problem determination for syslogd logging

If a log file does not contain any messages, first make sure the job is running. If the job is running, restart syslogd and add -u as an additional parameter. The -u parameter adds user ID/job name information to the message. Once syslogd is running again with the -u parameter, check the msgs file to determine if perhaps the job-specific log file is empty because the job name does not match the job name specified in /etc/syslog.conf. Example 1-15 shows a section of our msgs file where INETD and syslogd were started with the wrong job name.

Example 1-15 Entries in msgs file with incorrect job name

```
Sep 20 21:07:45 WTSC31/CS01 1  CS09 2 inetd[19]: FOMN0066 inetd terminating
Sep 20 21:09:33 WTSC31/CS01 1  CS06 3 FSUM1220 syslogd: restart
Sep 20 21:09:33 WTSC31/CS01 1  CS06 3 FSUM1232 syslogd: running non-swappable
```

The field immediately following the date appears as a result of the -u parameter and identifies the system/user ID 1 that started the job. The next field also appears as a result of the -u parameter and identifies the job name 2 3. The next field after the job name is the beginning of the message. As you can see in Example 1-15, inetd was incorrectly named CS09 2 and syslogd was incorrectly named CS06 3, which would explain why Example 1-15 shows that the inetd and syslogd log files are empty.

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

1.5 Additional information sources for syslogd

For additional information, refer to:

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

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Archived

Archived

2

TN3270E Telnet server

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

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

This chapter discusses the following topics.

Section	Topic
2.1, “Conceptual overview of the TN3270E server” on page 16	Basic concepts of TN3270E servers.
2.2, “TN3270E server in a single image” on page 22	A TN3270E basic implementation scenario in a single-image environment, including description, configuration, activation and verification.
2.3, “Multiple TN3270E servers in a multiple image environment” on page 52	A multiple TN3270E servers implementation scenario in a sysplex environment, including description, configuration, activation and verification.
2.4, “Problem determination for the TN3270E servers” on page 70	Problem determination methods for TN3270E server environments.
2.5, “Additional information sources for the TN3270E server” on page 77	References for TN3270E implementations.

© Copyright IBM Corp. 2008. All rights reserved.

2.1 Conceptual overview of the TN3270E server

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

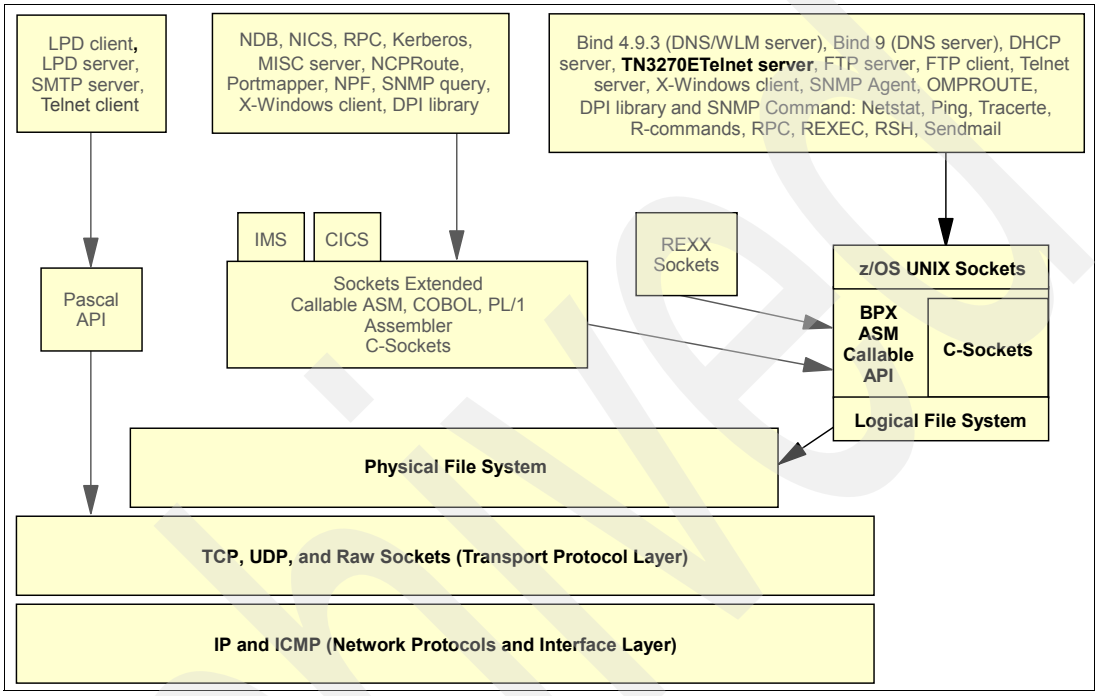


Figure 2-1 z/OS TN3270E Telnet server application services

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

This section discusses the following topics:

- What is the TN3270E server
- How does the TN3270E server work
- How can the TN3270E server be used

2.1.1 What is the TN3270E server

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

Traditional Telnet and TN3270 differ in two main areas:

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

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

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

327x device types that end in -E are terminals that support extended field attributes, such as color and highlighting. TN3270E server support includes a number of important enhanced capabilities over TN3270, and these are discussed in greater detail in “TN3270E functionality” on page 18.

For descriptions of security considerations that affect specific servers or components, refer to “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

2.1.2 How does the TN3270E server work

The TN3270 protocol provides an efficient means for transporting SNA 3270 traffic over an IP network while enabling the migration of user access to TCP/IP-only connectivity, and away from native SNA networking support. The TN3270E server acts as a VTAM application that activates one VTAM application minor node logical unit (LU) to represent each active TN3270 client connection.

Using TN3270 is a two-step process. First the client connects via TCP/IP to the TN3270E server that is listening on some TCP port (usually port 23). Then the TN3270E server allocates VTAM resources and establishes a logical session on behalf of the connecting client.

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

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

- TN3270E functionality
- Connection security
- Connection mapping

- Connection and session management
- Connection diagnostics

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

TN3270E functionality

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

Some of the features of TN3270E are illustrated in Figure 2-2 on page 18.

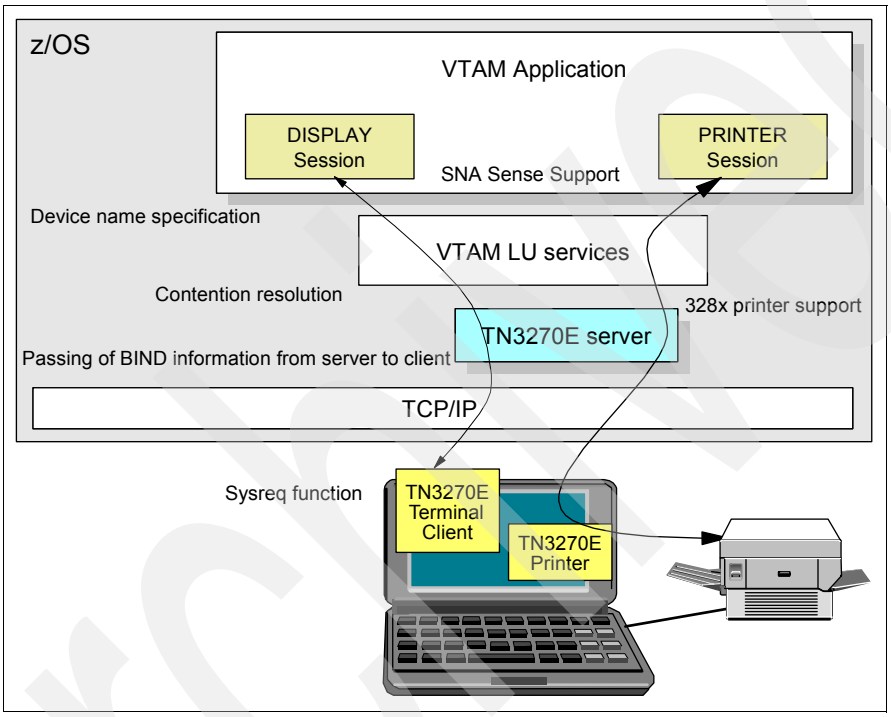


Figure 2-2 TN3270E negotiable features

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

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

Connection security

Important connection security functions include:

- Data overrun security
  - Protects against a client hung in a send-data loop
  - Protects against using large amounts of storage
  - Limits the number of session requests received by TN3270 in a 10-second period

  - Limits the number of chained RUs received without a corresponding end chain RU
  - Avoids auto-reconnect loops upon client connect errors by keeping connection active
- Network Access Control (NAC)
  - Limits user access to certain IP security zones defined by the NETACCESS statement.
  - Manages NAC user ID send and receive access for these security zones
  - NAC user ID is based on the application's address space information
- Transport layer security
  - Secures TN3270 connections with the transport layer security (TLS) or secure sockets layer (SSL) protocol
  - Enables installations to support both types of secure clients without knowing which protocol the client is using
  - SSLV2, SSLV3, and TLS supported (NOSSLV2 is the default.)
  - Enables using Application Transparent Transport Layer Security (AT-TLS) to manage secure connections for TN3270E.

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

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

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

Be aware of these AT-TLS capabilities and requirements when planning AT-TLS support for TN3270:

- Dynamically refresh a key ring.
- Support new or multiple key rings.
- Specify the label of the certificate to be used for authentication instead of using the default.
- Support SSL session key refresh.
- Support SSL session reuse.
- Support SSL sysplex session ID caching.
- Trace decrypted SSL data for Telnet in a data trace.
- Receive more granular error messages in syslogd for easier debugging.
- Policy Agent must be active.
- TLS security defined within the TN3270 profile will continue to be available and can be implemented concurrently with AT-TLS.

Refer to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535, for complete details about implementing TLS and AT-TLS security for the TN3270 server.

Note: The TN3270 client, activated through the TSO TELNET command, does not support any SSL or TLS security protocols. Basic mode is the only method supported by the TN3270 client.

TLS/AT-TLS recommendations

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

If, however, you do not need those additional functions, consider implementing FTP and TN3270 with the full AT-TLS support. AT-TLS is the preferred method of implementing TLS support in order to achieve a consistent solution for *all* of your TCP applications.

Connection mapping

When a client connection request is made, TN3270 must assign an LU name to represent the client. Additionally, an Unformated System Services (USS) table, an interpret table, a default application, unique parameters, and network monitoring actions can optionally be assigned to the connection. There are 11 mapping statements available in the BEGINVTAM block that map objects to specified client identifiers within the port indicated on the BEGINVTAM block. This robust mapping function gives you flexibility in supporting your organization's requirements for running TN3270 services on the z/OS platform.

Connection and session management

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

- Session initiation management:
  - Use of VTAM interpret tables
  - Use of VTAM USS tables
  - Use of VTAM default applids
- Connection and session takeover: assists with lost, disconnected sessions.
- Queuing sessions: assists with session manager applications.
- Disconnect on session error: determines type of session termination.
- Bypass RESTRICTAPPL with CERTAUTH: client certificate is used to derive a user ID.
- Allow printer sessions with RESTRICTAPPL: enables printers to establish a session with an application defined as a RESTRICTAPPL, because printers cannot submit a user ID and password.
- The option of keeping the ACB open for an application that needs to INQUIRE for status.
- Express Logon Feature (ELF): TN3270 uses the client certificate to resolve the user ID and RACF® generates a temporary password, a passticket. ELF requires a secure connection with level 2 client authentication, a client that supports ELF, and RACF passticket setup.
- Cleanup on idle connections: Use the PROFILEINACTIVE parameter statement with a specified time-out value to drop connections associated with a non-current profile that do not have an SNA session.

Multilevel security compliance

The standalone TN3270E server can now participate in a multilevel secure environment that uses security labels from z/OS Communications Server. To ensure correct security label comparisons, Network Access Control (NAC) must also be active for Telnet. For more information about preparing for TCP/IP networking in a multilevel secure environment and NAC, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Connection diagnostics

In addition to general diagnostic tools such as CTRACE and dumps, described in *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, and *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532), there are several TN3270-specific diagnostic tools available:

- Debug messages: TN3270-specific debug messages can be turned on or off to diagnose TN3270 problems. Several types of debug messages are available.
- Summary messages indicate important status changes.
- Detail messages indicate that a problem was detected.
- General messages indicate an important event.
- Trace messages show data to and from the client, and to and from VTAM.
- Flow messages show entry into modules and exit from modules.
- MSG07 enables TN3270 to send a message to the client indicating an error occurred and what the error was.
- The ABENDTRAP command can be used to set up an abend based on the variables specified.
- Optional SMF recording is controlled by using the SMFINIT and SMFTERM statements.
- TESTMODE causes the profile syntax to be checked without making the profile active.
- Several displays are available that provide summary and detail information.
- TCP/IP Packet Trace using the system's CTRACE facility (SYSTCPDA).
- TCP/IP Socket Data Trace using the system's CTRACE facility (SYSTCPDA).
- TCP/IP Component Trace using the system's CTRACE facility (SYSTCPIP).

Refer to "Problem determination for the TN3270E servers" on page 70 for more detailed examples.

2.1.3 How can the TN3270E server be used

This section discusses the following TN3270 scenarios:

- TN3270E server in a single image
- Multiple TN3270E servers in a multiple image environment

These two scenarios are commonly implemented on single-image and sysplex environments. For other advanced TN3270 high availability or security implementation scenarios, refer to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534, and to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535.

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

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

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

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

See Appendix B, "Sample files provided with TCP/IP" on page 323, for sample files available for use with the TN3270E server.

2.2 TN3270E server in a single image

This section provides an overview of executing the TN3270E server in one MVS image.

The following topics are discussed:

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

2.2.1 Description of our TN3270E server scenario

In this scenario we use one system image, one TCP/IP stack, and one TN3270E server running in its own address space. Stack affinity is established for the TN3270E server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block. The relationship between the TN3270E server, the TN3270 client, and the TCP/IP stack is illustrated in Figure 2-3.

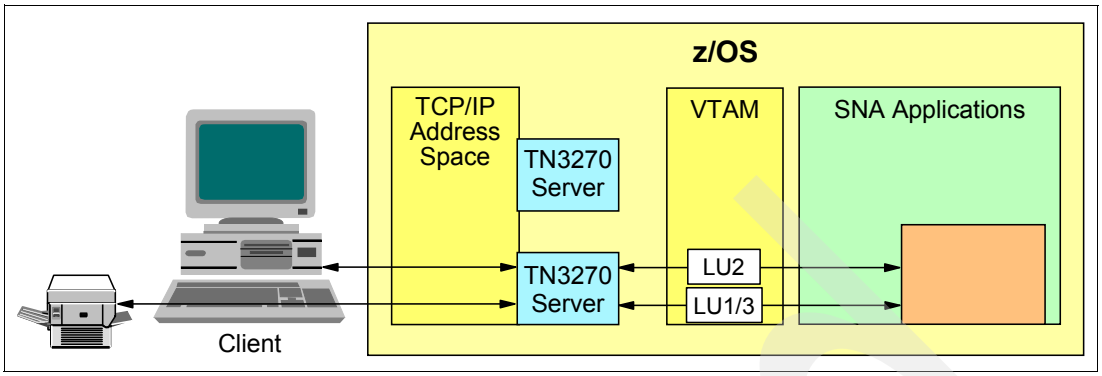


Figure 2-3 TN3270E Telnet server executing in a MVS image

As shown in Figure 2-3, you can run one or multiple TN3270E servers in a single-image system, each in its own address space. In this section, we discuss the implementation of one TN3270 started task. In the same way, you can set up multiple separated TN3270E servers in one LPAR.

Our TCP/IP stack started task name is TCPIP. We use system symbolics in the started task JCL to provide uniqueness where necessary. The TN3270E started task name is TN3270.

2.2.2 Configuration of the TN3270E server

Perform the following tasks to configure the TN3270E server:

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

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

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

Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for an in-depth discussion about how to prepare the VTAM LU definitions for the TN3270E server. Attention should be given to the values coded for the following keywords on the VTAM APPL statement:

- SESSLIM=
- EAS=
- PARSESS=

Some SNA applications do not issue CLSDST when their LOSTERM exit is driven. This could create a hang condition for the TN3270 LU that has issued a CLOSEACB and is waiting for an UNBIND RESPONSE from the application.

Note: Code LOSTERM=IMMED on all target (PLU) applications that will have a SNA session with TN3270 to avoid CLOSEACB hang conditions. Code EAS=1 to minimize Common Service Area (CSA) storage use. Code PARSESS=NO for parallel sessions should not be used with Telnet LUs. Code SESSLIM=YES because Telnet server LUs do not support multiple concurrent sessions.

These Logical Unit (LU) names represent each TN3270 client IP connection. It is this VTAM LU that logs on to a selected VTAM application. The TN3270E server uses application LUs that are defined in VTAM application (APPL) major nodes to represent clients, by making them look and act like VTAM terminal LUs. These APPL definition members must be made available to VTAM by being in the list of data sets specified on the VTAMLST DD statement in the JCL procedure used to start VTAM. This VTAM major node member contains the TN3270 application LUs, and ensures that these LUs will be available for activation after VTAM, TCP/IP, and the TN3270E server are started. An APPL statement can be entered for each LU or a model APPL statement can be used. You should use a model statement in order to avoid all the clerical effort of maintaining a large list. There is an extensive discussion on coding techniques for the model APPL statement in *z/OS Communications Server: SNA Resource Definition Reference*, SC31-8778. Our model statement uses an asterisk (*) in the name as a wild card. You can also use system symbolics to assist when multiple systems are involved and you want them to share the VTAMLST and the same member within that VTAMLST. One of the techniques for using system symbolics is shown in Example 2-1.

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

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

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

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

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

Add the VTAM APPL major node to the VTAM startup configuration

To automatically activate the application definition major node, include it in ATCCONxx. If multiple TN3270E servers are used (for example, multiple TCP/IP stacks in a sysplex environment), ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail, or the cross-domain session request will fail.

Customize the TCPDATA configuration data set

This file is normally customized during basic stack setup and initialization. For details on customizing the TCPDATA file and the resolver see *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532. TN3270 uses the native MVS sockets search order to find a resolver. Neither the Resolver_Config environment variable nor the /etc/resolv.conf is used by TN3270 when searching for TCPIP.DATA. If no resolver has been defined, no host name will be found.

Design the mapping of APPL and LU assignments

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

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

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

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

The NULL set of clients is defined as that group of clients for which no explicit mapping statement applies. Mapping statements that do not require an assignment to a specific client ID can be used to set the default assignments for the NULL client group, by simply omitting the client ID from the statement. These mapping statements can omit the client ID: DEFAULTAPPL, PRTDEFAULTAPPL, LINEMODEAPPL, USSTCP, and INTERPTCP. There are additional statements whose very names imply a mapping association to the NULL client group: DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTPRT, and DEFAULTPRTSPEC. These uniquely named groups do not require mapping, they imply it.

For complete details on mapping objects to client identifiers, refer to that topic in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. Complete statement syntax is shown in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. These two books provide comprehensive discussions and examples related to mapping. We do not intend to duplicate the discussions or examples in this book. The following discussion assists you with comprehending the text in the books referenced.

There are 15 object types that can be mapped to 10 client ID types. There are 9 object types that can be mapped to the NULL client ID. Therefore, the total number of unique mapping combinations supported is 159. It is not practical to attempt to show an example of each one of these mappings. However, we do want to discuss a methodology you can use to define a valid mapping strategy. Obviously, the scenario that yields the least amount of clerical effort in customizing a TN3270 configuration file is the one that does not require any explicit mapping assignments. In many cases, a TN3270E server can assign all connecting clients to an LU

name from the DEFALUTLUS pool, and force them all to the same DEFAULTAPPL or to the same USSTCP table. If this meets your organization's needs, then customizing the TN3270 profile data set will be minimal effort for you.

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

Understand connection mapping terminology

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

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

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

However, groups must be defined prior to referring to them in a mapping statement. This applies to both object groups and client ID groups. We define a few terms in Table 2-1 on page 26.

Table 2-1 Group types used in mapping statements

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

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

Example 2-4 Mapping statement format

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

Adopt a connection mapping methodology

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

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

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

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

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

Tip: In order to establish a consistent grouping methodology, consider using this technique:

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

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

Customize the TN3270 PROFILE data set

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

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

The purpose of the TN3270 configuration statements is to:

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

The TN3270 configuration uses the following statement blocks:

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

Stack affinity issues

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

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

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

In a common INET (CINET) environment, TN3270 is, by default, associated with all running stacks. If another stack is started while TN3270 is active, the current LISTEN for the port is cancelled and reissued automatically to include the new stack. If association with one stack is desired for control purposes or for functionality support, specify TCPIPJOBNAME.

TN3270 SNMP subagent limitation

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

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

SMF address space name field

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

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

With the Telnet server within the stack, they are accustomed to seeing the stack task name. Now when they change to its own address space, there will be a new name in the field, and the code may have to be changed.

Port reservation

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

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

```
PORT 23 TCP TN3270 NOAUTOLOG
```

A portion of the TN3270 configuration profile data set is shown in Example 2-5. Refer to Appendix C, “Configuration files: TN3270E standalone started task scenario” on page 331, to review the complete profile.

Example 2-5 TN3270 configuration profile (TELNB30A) for standalone task

```
BROWSE    TCPIP.TCPPARMS(TELN30A) - 01.02           Line 00000000 Col 001 080
; ===SC30============ TN3270E Telnet server Profile for Standalone Task =======-
; No SSL security.  No sysplex distribution in the stack.            -
;        This is a plain server profile, very simple and easy.       -
; ----------------------------------------------------------------------
TELNETGLOBALS
. . . . . . . . . .
    TCPIPJOBNAME TCPIP
. . . . . . . . . .
ENDTELNETGLOBALS
;
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
```

For complete detailed listings of the started task procedures and profiles used for this scenario, see Appendix C, “Configuration files: TN3270E standalone started task scenario” on page 331.

Define system security for the TN3270 started task

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

This discussion assumes RACF is the security subsystem being used. If another security product is used, refer to its publications for equivalent setup instructions. When TN3270 executes in its own address space, set up for the started task is necessary. Before TN3270 can be started, security for the procedure name and its associated user ID must be defined. Refer to the TN3270 chapter in *z/OS Communications Server: IP Configuration Guide*,

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

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

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

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

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

```
ADDUSER TN3270
ALTUSER TN3270 OMVS(UID(0) PROGRAM ('/bin/sh') HOME('/'))
```

In this example the user ID name is TN3270, but any name can be used. These two RACF commands can be combined into one command by putting the OMVS parameter on the ADDUSER command line. The add and alter commands are done separately in case the user ID already exists. If the add fails, the alter still succeeds.

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

- Add the user to RACF:

  ```
  ADDUSER TN3270
  ```

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

    ```
    RDEFINE FACILITY BPX.SUPERUSER
    ```

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

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

  - Permit the user to the class:

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

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

  - Refresh the FACILITY class:

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

Consider program properties table attributes for the TN3270E server

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

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

```
EZZ0621I AUTOLOG FORCING TN3270, REASON: AUTOLOGGING SCHEDULED
IEE838I TN3270 NON-CANCELABLE - ISSUE FORCE ARM
```

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

```
PORT 23 TCP TN3270 NOAUTOLOG
```

Customize the JCL for the TN3270 started task

A sample JCL can be found in SEZAINST(EZBTNPRC).

The only valid parameter that can be passed in from the JCL is the component trace options parmlib member name.

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

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

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

Example 2-6 JCL for the TN3270E server: TN3270

```
BROWSE    SYS1.PROCLIB(TN3270) - 01.00            Line 00000000 Col 001 080
//TN3270 PROC PARMS='TRC=TN'
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//TN3270 EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIP.TCPPARMS(TELNB&SYSCLONE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.TCPPARMS(DATAB&SYSCLONE.)
```

2.2.3 Activation of the TN3270E server

To start the Tn3270E started task, issue the MVS START command or automate it with an automation package:

```
S TN3270
```

Expect to see initialization messages similar to these:

```
IEF695I START TN3270   WITH JOBNAME TN3270   IS ASSIGNED TO
$HASP373 TN3270   STARTED
ACF9CCCD USERID TN3270   IS ASSIGNED TO THIS JOB - TN3270
IEF403I TN3270 - STARTED - TIME=04.11.06
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
```

2.2.4 Verification of the TN3270E server

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

This section discusses the following topics:

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

Using HELP for Telnet commands

Telnet has created its own Help commands and Stor command to replace the TCP/IP versions. Any Telnet command submitted to the TCP/IP address space is ignored and message EZZ0210I is issued to indicate it. Following is the Telnet Help command's syntax:

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

Telnet Help commands are used to see the format of any Telnet Display or Vary command. Each help command will show the options available at the next level of detail. For example, if you issue D TCPIP,tnproc,T,HELP you will see that you can issue help for either STOR or TELNET. If you then issue D TCPIP,tnproc,T,HELP,TELNET you will see all the display Telnet options available.

The Telnet Stor command displays the current and maximum storage usage status and identifies the service level of Telnet modules in the previous release, as shown here:

```
D TCPIP,TN3270,TELNET,STOR
EZZ8453I TELNET STORAGE
EZZ8454I TN3270   STORAGE    CURRENT   MAXIMUM     LIMIT
EZZ8455I TN3270   ECSA           65K       85K   NOLIMIT
EZZ8455I TN3270   POOL         7619K     7853K   NOLIMIT
EZZ8455I TN3270   CTRACE     262372K   262372K   262372K
EZZ8459I DISPLAY TELNET STOR COMPLETED SUCCESSFULLY
```

Contrasted with the TCPIP Stor command's output, the Telnet Stor command displays the storage usage status for CTRACE. TCPIP CTRACE storage resides in a separate dataspace and is not part of the storage display, but Telnet CTRACE resides in Telnet's private address space. If only the total POOL value were shown, the CTRACE amount would obscure the amount of storage used by Telnet processes. Therefore, before the data is presented, the CTRACE amount is subtracted from the total POOL amount and presented on its own line.

The CTRACE amount appears large because Telnet always allocates a 256 M block of storage to support the largest CTRACE BUFSIZE amount. This storage is not wrapped until it

is filled in with data. You will never use real storage resources for more than the amount you define on the CTRACE BUFSIZE parameter.

When you forget to state the Telnet server procedure name on the V TCPIP,tnproc,T,OBEYFILE command to update your profile, message EZZ0209I is issued to indicate `TELNET SERVER CONFIGURATION STATEMENTS IGNORED IN TCPIP.`, instead of being processed by the default TCP/IP stack.

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

Using OBEYFILE to modify a configuration

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

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

- After VARY TCPIP,,OBEYFILE command processing completes, the new profile is labeled the CURRent profile, and the replaced profile becomes profile 0001. After another update, the new update becomes the current profile and the replaced profile becomes profile 0002. If the profile update is for a subset of the active ports, the ports not being updated remain unchanged. Profile debug messages can be suppressed by coding DEBUG OFF or DEBUG SUMMARY in TELNETGLOBALS and placing it before all other Telnet statement blocks.
- New connections are associated with the current profile and use the mappings and parameters defined by that profile. Even if a VARY TCPIP,,OBEYFILE command updates the port, existing connections remain associated with the same profile. The statements of non-current profiles remain in effect and continue to support all connections that were established when the non-current profile was the CURRent profile. When all connections associated with a non-current profile have ended, the storage for the non-current profile mapping rules is freed and the profile is considered INACTIVE.

Using OBEYFILE and TESTMODE to syntax check the profile statements

To validate a TN3270 profile without applying the profile, specify TESTMODE (a TELNETPARMS-only parameter). When no errors are reported, remove the TESTMODE statement. A sample of the TESTMODE statement is shown in Example 2-7.

Example 2-7 TESTMODE statement in TN3270 profile

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

Use the OBEYFILE command to validate the new profile and also to activate the new profile. The OBEYFILE command is entered the same either way. The TESTMODE statement makes the difference between validation and actual activation. The OBEYFILE command is shown in Example 2-8.

Example 2-8 OBEYFILE command to validate or activate a TN3270 profile

```
V TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNB30A)
```

Example 2-9 shows the resulting messages from the OBEYFILE command when the TESTMODE statement is *included* in the TN3270 profile.

Example 2-9 Messages issued when TESTMODE is included in the TN3270 profile

```
V TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNB30A)
   EZZ0060I PROCESSING COMMAND: VARY TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNB30A)
   EZZ0300I OPENED OBEYFILE FILE 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ0400I TELNET/VTAM (SECOND PASS) BEGINNING FOR FILE: 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ6018I TELNET PROFILE TESTMODE COMPLETE FOR PORT    23            1
   EZZ0403I TELNET/VTAM (SECOND PASS) COMPLETE FOR FILE: //'TCPIP.TCPPARMS(TELNB30A)'
```

1 Message EZZ6018I indicates that the TESTMODE scan has been performed and no activation of the profile has taken place

Example 2-10 shows the resulting messages from the OBEYFILE command when the TESTMODE statement was *removed* from the TN3270 profile.

Example 2-10 Messages issued when TESTMODE is removed from the TN3270 profile

```
V TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNB30A)
   EZZ0060I PROCESSING COMMAND: VARY TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNB30A)
   EZZ0300I OPENED OBEYFILE FILE 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ0400I TELNET/VTAM (SECOND PASS) BEGINNING FOR FILE: 'TCPIP.TCPPARMS(TELNB30A)'
   EZZ6003I TELNET LISTENING ON PORT    23                            1
   EZZ0403I TELNET/VTAM (SECOND PASS) COMPLETE FOR FILE: //'TCPIP.TCPPARMS(TELNB30A)'
```

1 Message EZZ6003I indicates that the profile has been activated and the TN3270 server is listening on the specified port.

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

Checking maximum connections supported

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

Checking the total number of TN3270 ports defined in the profile

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

Using DISPLAY commands to view the status of TN3270 resources

For all commands referred to in this section, see:

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

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

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

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

Most of the TELNET Display commands are grouped into one of the following categories:

- D TCPIP,,TELNET,PROF,....
- D TCPIP,,TELNET,CLID,....
- D TCPIP,,TELNET,OBJ,....
- D TCPIP,,TELNET,CONN,....

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

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

Using display PROFILE to view profile information

The PROFILE display command enables you to determine what profile-wide options are in effect for each profile, which profiles are being used, and how many users are on each profile. In the next few display examples, notice the difference between summary output and detailed output. And because port 23 is defined as a BASIC (non-secure) port, any display command specifying SECURE information will result in a "no matches found" condition. Fields of interest to this discussion are *highlighted* in the output messages. Because port 23 is a BASIC port (and the only BASIC port), the results of specifying BASIC or omitting it are the same. A profile summary report shows a summary of the port settings in Example 2-11.

Example 2-11 Display PROF, SUM shows profile summary for all profiles

```
D TCPIP,TN3270,T,PROF,SUM
   EZZ6060I TELNET PROFILE DISPLAY 052
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      0
   ------------------------------------------------------------
      FORMAT            LONG
   . . .
```

A profile detail report shows the port settings and includes a legend to assist in interpreting the abbreviated settings, as seen in Example 2-12.

Example 2-12 Display PROF, DET shows profile detail for all profiles

```
D TCPIP,TN3270,T,PROF,DET
   EZZ6080I TELNET PROFILE DISPLAY 054
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
     -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
     LM-----  ---S--------  --F  -B--*----  *---ST*  S--- *TPARMS
     LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* CURR
    PERSISTENCE
      LUSESSIONPEND
      MSG07
      ....
    FUNCTIONS
      TN3270E
      SNAEXTENT
      ....
    DIAGNOSTICS
       DEBUG             EXCEPTION
       DEBUG ROUTING     CONSOLE
       FULLDATATRACE
   SECURITY
       PORT                  23
       CONNTYPE          BASIC
       KEYRING           **N/A**
       CRLLDAPSERVER     **N/A**
       ENCRYPTION        **N/A**
       CLIENTAUTH        **N/A**
       NOEXPRESSLOGON
       NONACUSERID
       NOSSLV2
     TIMERS
       .....
   MISCELLANEOUS
     ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      0
     ------------------------------------------------------------
       FORMAT            LONG
       TCPIPJOBNAME      TCPIP
       TNSACONFIG        DISABLED
    . . .
```

The profile report can be limited to BASIC ports only, as seen in Example 2-13.

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

```
D TCPIP,TN3270,T,PROF,PROF=BASIC,SUM
   EZZ6060I TELNET PROFILE DISPLAY 056
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      0
   ------------------------------------------------------------
      FORMAT            LONG
      TCPIPJOBNAME      TCPIP
      TNSACONFIG        DISABLED
   . . .
```

The profile report can be limited to SECURE ports only, as seen in Example 2-14. Because there are no secure ports defined in this profile, message EZZ6057I indicates there are no entries to list.

Example 2-14 Display PROF, SECURE, SUM shows profile summary for secure profiles

```
D TCPIP,TN3270,T,PROF,PROF=SECURE,SUM
   EZZ6060I TELNET PROFILE DISPLAY 060
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

Using display CLIENTID command to view client ID information

The CLIENTID display can be used to see what client IDs are defined in the profile and details about the client IDs. In the next few display examples, notice the difference between summary output and detailed output. And because port 23 is defined as a BASIC (non-secure) port, any display command specifying SECURE information will result in a “no matches found” condition. Fields of interest to this discussion are *highlighted* in the output messages. Because port 23 is a BASIC port (and the only BASIC port), the results of specifying BASIC or omitting it are the same. A ClientID summary report shows any client groups you may have defined, as seen in Example 2-15.

Example 2-15 Display CLID, SUM shows clientid summary for all profiles

```
D TCPIP,TN3270,T,CLID,SUM

   EZZ6082I TELNET CLIENTID LIST 071
   USERID
     NO CLIENT IDS
   HOSTNAME
     NO CLIENT IDS
   IPADDR
     NO CLIENT IDS
   USERGRP
     NO CLIENT IDS
   HNGRP
     NO CLIENT IDS
   IPGRP
     NO CLIENT IDS
   DESTIP
     NO CLIENT IDS
   LINKNAME
     NO CLIENT IDS
   DESTIPGRP
     NO CLIENT IDS
```

```
   LINKGRP
     NO CLIENT IDS
   NULL
     NULL
   ----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   24 OF 24 RECORDS DISPLAYED
```

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

A ClientID detail report shows the how many connections are associated with which client ID groups, as seen in Example 2-16.

Example 2-16 Display CLID, DET shows clientid detail for all profiles

```
D TCPIP,TN3270,T,CLID,DET
   EZZ6081I TELNET CLIENTID DISPLAY 076
   CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
   NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
   ------------------ ------ --------- -------- ---------- --------
   NULL
     NULL
                          1 DEFAPPL   SC30TS               --------
   ----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   5 OF 5 RECORDS DISPLAYED
```

The ClientID report can be limited to BASIC ports only, as seen in Example 2-17 on page 38.

Example 2-17 Display CLID, BASIC, SUM shows client ID summary for basic profiles

```
D TCPIP,TN3270,T,CLID,PROF=BASIC,SUM
   EZZ6082I TELNET CLIENTID LIST 078
   USERID
     NO CLIENT IDS
   HOSTNAME
     NO CLIENT IDS
   IPADDR
     NO CLIENT IDS
   USERGRP
     NO CLIENT IDS
   HNGRP
     NO CLIENT IDS
   IPGRP
     NO CLIENT IDS
   DESTIP
     NO CLIENT IDS
   LINKNAME
     NO CLIENT IDS
   DESTIPGRP
     NO CLIENT IDS
   LINKGRP
     NO CLIENT IDS
   NULL
     NULL
   ----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
```

The ClientID report can be limited to SECURE ports only. Because there are no SECURE ports defined in the profile, message EZZ6057I indicates there are no entries to list, as seen in Example 2-18.

Example 2-18 Display CLID, SECURE, SUM shows client ID summary for secure profiles

```
D TCPIP,TN3270,T,CLID,PROF=SECURE,SUM
   EZZ6081I TELNET CLIENTID DISPLAY 082
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

Using display OBJECT command to view object information

The OBJECT display can be used to see what objects are defined in the profile and some details about the objects. In the next few display examples, notice the difference between summary output and detailed output. And because port 23 is defined as a BASIC (non-secure) port, any display command specifying SECURE information will result in a “no matches found” condition. Fields of interest to this discussion are *highlighted* in the output messages. Because port 23 is a BASIC port (and the only BASIC port), the results of specifying BASIC or omitting it are the same. The object summary report shows any objects that may be defined in the profile and what their assignments are, as seen in Example 2-19.

Example 2-19 Display OBJ, SUM shows object summary for all profiles

```
D TCPIP,TN3270,T,OBJ,SUM
   EZZ6084I TELNET OBJECT LIST 084
   ARAPPL
     SC*       NVAS*     TSO*      *
   DEFAPPL
     SC30TS
   PRTAPPL
     NO OBJECTS
   LINEAPPL
     NO OBJECTS
   MAPAPPL
     NO OBJECTS
   USS
     NO OBJECTS
   INT
     NO OBJECTS
   LU
     NO OBJECTS
   LUGRP
     *DEFLUS*
   APPLLUG
     NO OBJECTS
   PRT
     NO OBJECTS
   PRTGRP
     NO OBJECTS
   PARMSGRP
     *DEFAULT  *TGLOBAL  *TPARMS
   MONGRP
     NO OBJECTS
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   30 OF 30 RECORDS DISPLAYED
```

Archived

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

The object detail report indicates how many connections there are for each object group, as seen in Example 2-20.

Example 2-20 Display OBJ, DET shows object detail for all profiles

```
D TCPIP,TN3270,T,OBJ,DET
   EZZ6083I TELNET OBJECT DISPLAY 086
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   ARAPPL
    SC*            0
                                                            -A------
    NVAS*          0
                                                            -A--Q---
    TSO*           0
                                                            -A-D----
    *              1
                                                            -A------
   DEFAPPL         0
                                                            -I------
   DEFAPPL
    SC30TS         1 NULL      NULL
                                                            --------
   LUGRP
    *DEFLUS*       1
                                                            --------
   PARMSGRP
    *DEFAULT          -------NO MAPPING---------
                                                            --------
    *TGLOBAL          -------NO MAPPING---------
                                                            --------
    *TPARMS           -------NO MAPPING---------
                                                            --------
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   26 OF 26 RECORDS DISPLAYED
```

The object report can be limited to basic or to secure ports. Because there are no secure ports defined in this profile, message EZZ6057I indicates there are not entries to list, as seen in Example 2-21.

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

```
D TCPIP,TN3270,T,OBJ,PROF=SECURE,SUM
   EZZ6083I TELNET OBJECT DISPLAY 092
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

Using display CONN command to view connection information

The CONNECTION display command without the CONN= parameter gives you a high-level view of what connections exist and what they are being used for. A connection report shows all existing connections to each active port, as seen in Example 2-22.

Example 2-22 Display CONN shows all connections to the Telnet server

```
D TCPIP,TN3270,T,CONN,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 094
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   000000E8    ::FFFF:10.1.1.40..1028
                                      SC30BB01 SC30TS05  TA3 SNX32704
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
4 OF 4 RECORDS DISPLAYED
```

The connection report can be limited to basic ports only, showing only basic connections, as seen in Example 2-23.

Example 2-23 Display CONN, BASIC shows the basic connections only

```
D TCPIP,TN3270,T,CONN,PROF=BASIC,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 096
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   000000E8    ::FFFF:10.1.1.40..1028
                                      SC30BB01 SC30TS05  TA3 SNX32704
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The connection report shows secure ports and secure connections, as shown in Example 2-24 on page 41.

Example 2-24 Display CONN, SECURE shows the secure connections

```
D TCPIP,TN3270,T,CONN,PROF=SECURE,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 098
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and SUM parameter gives a *summary* look at one single connection, as seen in Example 2-25.

Example 2-25 Display CONN, CONN=, SUM shows summary information for one connection only

```
D TCPIP,TN3270,T,CONN,CONN=000000E8,SUM
   EZZ6064I TELNET CONNECTION DISPLAY 100
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   000000E8    ::FFFF:10.1.1.40..1028
                                      SC30BB01 SC30TS05  TA3 SNX32704
   ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and DET parameter gives a *complete* look at one single connection, as seen in Example 2-26.

Example 2-26 Display CONN, CONN=, DET shows detail information for one connection only

```
D TCPIP,TN3270,T,CONN,CONN=000000E8,DET
```

```
EZZ6065I TELNET CONNECTION DISPLAY 102
  CONNECTED: 23:13:55  09/24/2007  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 000000E8   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.1.30..1028
    DESTIP..PORT: ::FFFF:10.1.1.10..23
    LINKNAME: VIPA1L
  PORT:    23 QUAL: NONE
    AFFINITY: TCPIP
    STATUS: ACTIVE  BASIC         ACCESS: NON-SECURE
  PROTOCOL: TN3270            DEVICETYPE: IBM-3278-4-E
    TYPE: TERMINAL GENERIC
    OPTIONS: -TET----   3270E FUNCTIONS: *N/A*
                       NEWENV FUNCTIONS: --
  LUNAME: SC30BB01
  APPL: SC30TS05
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32704  APPL SPECIFIED: SNX32704
  MAPPING TYPE:  CONN IDENTIFIER
                      OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  NL (NULL)
                     >*DEFLUS*                  --------
    DEFLT APPL: NL (NULL)
                      SC30TS                    --------
    USS TABLE:  **N/A**
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  LM-----  ---S--------  --F  -B--*----  *---ST*  S--- *TPARMS
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* TP-CURR
  LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD* <-FINAL
35 OF 35 RECORDS DISPLAYED
```

The CONNECTION display command with the LUNAME= parameter and SUM parameter gives a summary look at the connection assigned to that LU name, as seen in Example 2-27.

Example 2-27 Display CONN, LUNAME=, SUM shows summary information for one LU name only

```
D TCPIP,TN3270,T,CONN,LUNAME=SC30BB01,SUM
EZZ6064I TELNET CONNECTION DISPLAY 104
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- ---------------------- -------- --------  --- --------
000000E8    ::FFFF:10.1.1.40..1028
                                   SC30BB01 SC30TS05  TA3 SNX32704
----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
-------------------------------------------------------------
4 OF 4 RECORDS DISPLAYED
```

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

2.2.5 Administration and management of the TN3270E server

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

- Use VARY to quiesce, resume, and stop a TN3270 port
- Use VARY to change the status of TN3270 LUs
- Check client connection status
- LU name assignment user exit
- System Symbols for USS tables
- Queued session timer
- Performance monitoring data collection

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

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

- VARY TCPIP,,TELNET,QUIESCE a port to block any new connection requests but allow existing connections to continue activity.
- VARY TCPIP,,TELNET,RESUME a port to end the QUIESCEd state and allow new connection requests.
- VARY TCPIP,,TELNET,STOP a port to end connections on the port, and close the port, and discard all information for that port as though it were never defined.
- VARY TCPIP,,OBEYFILE, to start, restart, or change a port by updating the TN3270 profile.

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

The QUIESCE command is shown in Example 2-28.

Example 2-28 QUIESCE Port 23

```
V TCPIP,TN3270,T,QUIESCE,PORT=23
   EZZ6038I TELNET COMMAND QUIESCE 23 COMPLETE
   EZZ6003I TELNET QUIESCED ON PORT    23
```

To verify the quiesce command, a display profile command is shown in Example 2-29.

Example 2-29 Port status after QUIESCE command

```
D TCPIP,TN3270,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 114
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
   ----- PORT:    23  QUIESCED            PROF: CURR CONNS:      1
   ------------------------------------------------------------
      FORMAT            LONG
      TCPIPJOBNAME      TCPIP
   . . .
```

If a client attempts to connect while the port is quiesced, the request is rejected, as shown in Example 2-30. We directed a connection request toward the quiesced port using the TSO TELNET command.

Example 2-30 Connection request is rejected when port is quiesced

```
A TSO telnet command was directed to this port:
   TELNET 10.1.1.10
   EZA8256I Connecting to 10.1.1.10, port TELNET (23)
    EZA8262I Foreign host rejected the open attempt (8547)
    ***
```

The RESUME command is shown in Example 2-31.

Example 2-31 RESUME port 23

```
V TCPIP,TN3270,T,RESUME,PORT=23
   EZZ6038I TELNET COMMAND RESUME 23 COMPLETE
   EZZ6003I TELNET RESUMED ON PORT    23
```

To verify the resume command, a display profile command is shown in Example 2-32.

Example 2-32 Port status after RESUME command

```
D TCPIP,TN3270,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 141
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
   ----- PORT:    23  ACTIVE                PROF: CURR CONNS:      1
   ------------------------------------------------------------
      FORMAT           LONG
      TCPIPJOBNAME     TCPIP
   . . .
```

An example of the STOP command is shown in Example 2-33 on page 44. The client connection that was active is forced off and messages show the adverse affect on the connection. All connections on port 23 would be terminated.

Example 2-33 STOP port 23

```
V TCPIP,TN3270,T,STOP,PORT=23
   EZZ6038I TELNET COMMAND STOP 23 COMPLETE
   IKT100I USERID          CANCELED DUE TO UNCONDITIONAL LOGOFF
   IKT122I IPADDR..PORT 10.1.1.40..1028
EZZ6010I TELNET SERVER ENDED FOR PORT    23
```

To verify the stop command, a display profile command is shown in Example 2-34.

Example 2-34 Port status after STOP command

```
D TCPIP,TN3270,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 358
   EZZ6057I NO QUALIFYING MATCHES
0 OF 0 RECORDS DISPLAYED
```

Using the OBEYFILE command to restart port 23 is shown in Example 2-35.

Example 2-35 Restart port 23 with OBEYFILE command

```
V TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNB30A)
   EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 476
               TCPIP.TCPPARMS(TELNB30A)
```

```
   EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 477
               TCPIP.TCPPARMS(TELNB30A)
   EZZ6003I TELNET LISTENING ON PORT    23
   EZZ6038I TELNET COMMAND OBEYFILE  COMPLETE
```

To verify that port 23 is once again defined and active, a display profile command is shown in Example 2-36.

Example 2-36 Port status after restart with OBEYFILE command

```
D TCPIP,TN3270,T,PROF
   EZZ6060I TELNET PROFILE DISPLAY 571
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     LM*****  **TSBTQ***RT  ECF  BB*******  *P**ST*  SDD*
   ----- PORT:    23  ACTIVE               PROF: CURR CONNS:      0
   ------------------------------------------------------------
      FORMAT            LONG
      TCPIPJOBNAME      TCPIP
   . . .
```

Use VARY to change the status of TN3270 LUs

VARY TCPIP,,TELNET,ACT and VARY TCPIP,,TELNET,INACT LUs are for use by the Telnet server. If an LU is already in use, the INACT command fails. Specify the name ALL to activate all inactive LUs with one command. These commands have no effect on the VTAM state of the LU.

To show a list of inactive TN3270 LUs before any are inactivated, use the Display INACTLUS command as shown in Example 2-37.

Example 2-37 Display inactive TN3270 LUs before an INACT

```
D TCPIP,TN3270,T,INACTLUS
   EZZ6061I TELNET INACTLUS DISPLAY 578
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

An example of the INACT command to inactivate a TN3270 LU is shown in Example 2-38.

Example 2-38 INACT command to inactivate a TN3270 LU

```
V TCPIP,TN3270,T,INACT,SC30BB26
   EZZ6038I TELNET COMMAND INACT SC30BB26 COMPLETE
```

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

Example 2-39 Display inactive TN3270 LUs after an INACT

```
D TCPIP,TN3270,T,INACTLUS
   EZZ6061I TELNET INACTLUS DISPLAY 582
   INACTIVE LUS
                  SC30BB26
   1 OF 1 RECORDS DISPLAYED
```

The ACT command to activate a TN3270 LU is shown in Example 2-40.

Archived

Example 2-40 ACT command to activate a TN3270 LU

```
V TCPIP,TN3270,T,ACT,SC30BB26
   EZZ6038I TELNET COMMAND ACT SC30BB26 COMPLETE
```

To show a list of inactive TN3270 LUs, use the Display INACTLUS command, as shown in Example 2-41. Because we reactivated SC30BB33, it no longer shows as an inactive LU.

Example 2-41 Display inactive TN3270 LUs after an ACT

```
D TCPIP,TN3270,T,INACTLUS
   EZZ6061I TELNET INACTLUS DISPLAY 605
   INACTIVE LUS
   EZZ6057I NO QUALIFYING MATCHES
   0 OF 0 RECORDS DISPLAYED
```

Check client connection status

If the telnet server receives a new connection from a client IP address that already has one or more existing connections, the server will check the existing connections to make sure they are still up. If not, the previously existing connections are cleaned up immediately. This improves the situation in which a user thinks his session has failed and he starts a new session (see Figure 2-4).

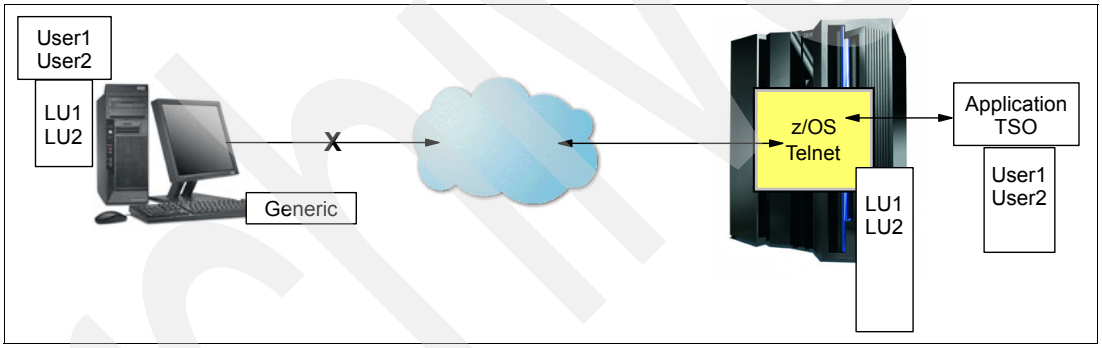


Figure 2-4 Client connection status

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

The **CheckClientConn** statement has the following syntax:

```
CheckClientConn sec,maxconn
```

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

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

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

Example 2-42 Example showing the definition for clients with the maximum of 60 telnet connections

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

Example 2-43 shows displaying the profile *before* adding the CheckClientConn statement. The default setting is indicated.

Example 2-43 Telnet profile without CheckClientConn enabled

```
D TCPIP,TN3270,TELNET,PROF,DETAIL
EZZ6080I TELNET PROFILE DISPLAY 714
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS  MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
  -------  ------------  ---  ---------  ------  ----
  *******  **TSBTQ***RT  EC*  BB*******  ***STS  *DD* *DEFAULT
  -------  ------------  ---  ---------  ------  ---- *TGLOBAL
  LM-----  --------W--T  ---  -B-------  *-*ST-  --L- *TPARMS
  LM*****  **TSBTQ*W*RT  EC*  BB*******  ***STS  *DL* CURR
 PERSISTENCE
   LUSESSIONPEND
   MSG07
   NOTKOSPECLU
   NOTKOGENLU
   NOCHECKCLIENTCONN
   NODROPASSOCPRINTER
   KEEPLU                 0 (OFF)
 FUNCTIONS
   NOOLDSOLICITOR
   NOSINGLEATTN
   TN3270E
   SNAEXTENT
   UNLOCKKEYBOARD BEFOREREAD
   UNLOCKKEYBOARD TN3270BIND
   SEQUENTIALLU
   NOSIMCLIENTLU
```

Example 2-44 shows displaying the profile *after* adding the CheckClientConn statement. The default setting is indicated.

Example 2-44 Telnet profile with the CheckClientConn enabled

```
D TCPIP,TN3270,TELNET,PROF,DETAIL
EZZ6080I TELNET PROFILE DISPLAY 663
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS  MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IKPSTS)(SMLT)
  -------  ------------  ---  ---------  ------  ----
  *******  **TSBTQ***RT  EC*  BB*******  ***STS  *DD* *DEFAULT
  -------  ------------  ---  ---------  ------  ---- *TGLOBAL
  LM--C--  --------W--T  ---  -B-------  *-*ST-  --L- *TPARMS    1
  LM**C**  **TSBTQ*W*RT  EC*  BB*******  ***STS  *DL* CURR
```

```
PERSISTENCE
  LUSESSIONPEND
  MSG07
  NOTKOSPECLU
  NOTKOGENLU
  CHECKCLIENTCONN         5,60
  NODROPASSOCPRINTER
  KEEPLU                  0 (OFF)
FUNCTIONS
  NOOLDSOLICITOR
  NOSINGLEATTN
  TN3270E
  SNAEXTENT
  UNLOCKKEYBOARD BEFOREREAD
  UNLOCKKEYBOARD TN3270BIND
  SEQUENTIALLU
  NOSIMCLIENTLU
  WLMCLUSTERNAME
```

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

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

LU name assignment user exit

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

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

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

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

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

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

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

System Symbols for USS tables

In recent releases it has been possible to use symbolic variables, such as luname, IP addresses, IP port, and hostname in the MSG10 definition. This has been expanded to include system symbols, such as the system name, system release and others.

An easy way to review potential symbols is with the MVS command D SYMBOLS to display the potential symbols that might be used in MSG10. Example 2-45 illustrates usage of system symbols.

Example 2-45 USS Table Source code with SYSNAME and SYSR9

```
 MSG10    DC      AL2(MSG10E-MSG10S)
 MSG10S   EQU     *
          DC      X'F5C31140401D40' ERASE/WRITE,WCC,SBA R1C1,
          DC C'WTSC30 You are connected to a non-SNA terminal - - - '
          DC      X'11C1501DE4'
          DC CL50'                                                  '
          DC CL30'                                  '
          DC CL50'System Name:  &&SYSNAME.                          '
          DC CL30'z/OS Release: &&SYSR9.            '
          DC CL50'                                                  '
          DC CL30'                                  '
.....................................
```

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

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

In Figure 2-5 on page 49 we can see the USStable hello screen presentation from the definitions that we used in Example 2-45.

```
WTSC30 You are connected to a non-SNA terminal - - -

 System Name:  SC30                          z/OS Release: Z1xRE9

                                @@@                 @@@  @@@@@@@@@
                                @@@@@@     @@@@          @@@ @@@@@@@@@@@@@@@@@@
International Technical        @@@@@@@@@@@@@@@          @@@@@@@@@@@@@@@@@@@@@@
Support Organization (ITSO)     @@@@@@@@  @@@        @ @ @ @@@@@@@@@@@@@@@@@@
                                @@@@@@@@@@@            @@@@@@@@@@@@@@@@@@@@@@
Poughkeepsie Center               @@@@@@@@@          @@@@@@@@@@@@@@@@@@@@@@@@
                                   @ @@@@  @          @@   @ @@  @@@@@@@@@@@ @@
                                    @@              @@@@@@ @@@@@@@@@@@@@@  @@
```

Figure 2-5 The USSTEST2 compiled including SYSNAME and SYSR9

The corresponding definition in the TCP/IP profile could be as shown in Example 2-46.

Example 2-46 TCP/IP profile mapping the new USS Table

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

- 1 USS Table for general purposes
- 2 USS Table mapped for a specific IP address

Queued session timer

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

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

Example 2-47 Display to see options for Allow/Restrict Appls

```
D TCPIP,TN3270,T,OBJ,TYPE=ARAPPL
EZZ6083I TELNET OBJECT DISPLAY 504
OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
----------  ------ --------- ---------------- ---------- --------
ARAPPL
 SC*            0
                                                          -A------  1
                                                        5 ----Q---  2
 TSO*           0
                                                          -A------
                                                        5 ----Q---
 *              0
                                                          -A------
                                                        5 ----Q---
DEFAPPL         0
                                                          -I------
----- PORT:    23  ACTIVE               PROF: CURR CONNS:      1
-------------------------------------------------------------
14 OF 14 RECORDS DISPLAYED
```

In Example 2-47 we can see the following:

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

Archived

Important: When using a Qsession Timer:

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

Performance monitoring data collection

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

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

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

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

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

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

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

Additional information about the Network Management Interface is available in *IP Programmer‘s Guide and Reference.* Two request options of particular interest are:

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

Filters for GetTnConnectionData are available, including the following:

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

Telnet SMF 119 SNA sessions termination records have changed with this release. Part of the Telnet section of the TCP connection termination record has a reason code. If the connection was closed by the TN3270E server the record contains the associated reason code. Refer to Appendix C, SMF type 119 records in *IP Configuration Reference* for more details.

Note that two SMF sections appear after the hostname triplet:

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

- Time bucket data

Note: Remember to configure Monitoring in the profile and create a MonitorGroup to map the group to clients using the MonitorMap statement.

You do not need to set up TNSACONFIG if SNMP is not being used.

Real-time SMF information service access control

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

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

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

2.3 Multiple TN3270E servers in a multiple image environment

This section provides an overview of executing multiple TN3270E servers in a sysplex environment with a server on each LPAR. This environment is illustrated in Figure 2-6.

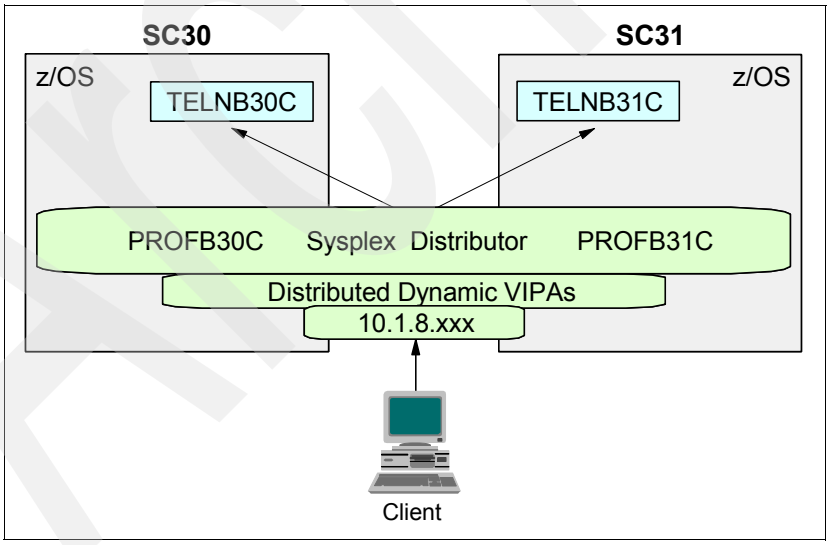


Figure 2-6 TN3270E server within the sysplex

In Figure 2-6, clients connect to the Distributed DVIPA address of the TN3270E server. Based on installation policies, the Sysplex Distributor (SD) will then direct connections to the best available TN3270E server.

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

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

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

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

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

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

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

2.3.1 Description of multiple TN3270E servers within the sysplex

This scenario is based on the TN3270E server scenario discussed in 2.2, “TN3270E server in a single image” on page 22.

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

For this scenario we use two system images: SC30 and SC31. Each system uses one TCP/IP stack. The TCP/IP stack started task name is the same on both systems: TCPIP.

Each stack has one TN3270E server associated with it. The TN3270E server started task name is the same on both systems: TN3270. Stack affinity has been established for the server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block. We use system symbolics in the started task JCL to provide uniqueness where necessary.

As Figure 2-6 on page 52 shows, on each system (SC30 and SC31) we have these two started tasks:

- TCPIP
- TN3270

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

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

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

TCP/IP stack redundancy

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

Stack dependencies for multiple TN3270E servers within the sysplex

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

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

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

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

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

TN3270E server redundancy

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

TN3270E server dependencies for multiple servers within the sysplex

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

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

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

> **Note:** TN3270 LU names to be used with VTAM must be unique for *each* server.
>
> Do *not* use the same LU names in multiple TN3270E servers.

Sysplex distribution

When you have multiple stacks with multiple TN3270E servers, as suggested previously, and are leveraging Parallel Sysplex technology, we recommend that you implement sysplex distribution in order to provide TN3270 connection load balancing at some level: round-robin, basewlm, or serverwlm. For detailed scenarios that cover these levels of load balancing, refer to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534.

2.3.2 Configuration of multiple TN3270E servers within the sysplex

In addition to the configuration tasks for every TN3270E server instance stated in "Configuration of the TN3270E server" on page 23, the following tasks are necessary to configure multiple TN3270E servers within the sysplex:

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

Customize a second TCP/IP stack started task procedure

This second started task is for running our sysplex distribution backup stack. It can be modeled after our first (primary) started task. We used the same proc name and same proclib for both stacks, and used system symbolics to provide unique names for the stacks' configuration profile data sets.

See Appendix D, "Configuration files: Multiple TN3270E Telnet servers and SD scenario" on page 343, for the started task procedures used for this scenario.

Customize a second TCP/IP stack configuration profile data set

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

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

See Appendix D, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 343, for the basic stack profile used for this scenario.

Customize a second TN3270E server started task procedure

This second started task is for running our second TN3270E server. It can be modeled after our first server started task. We used the same proc name and same proclib for both servers, and used system symbolics to provide unique names for the stacks’ configuration profile data sets.

See Appendix D, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 343, for the started task procedures used for this scenario.

Customize a second TN3270E server configuration profile

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

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

For a complete discussion and examples of setting up a TN3270E server, refer to 2.2.2, “Configuration of the TN3270E server” on page 23.

See Appendix D, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 343, for the basic TN3270E server profiles used for this scenario.

Establish an IP subnet for XCF interfaces

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

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

Establish an IP subnet for Dynamic VIPA

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

```
Our Distributed D-VIPA subnet is 10.1.8.0/24
Our Viparange   D-VIPA subnet is 10.1.9.0/24
```

Enable the primary stack to support sysplex functions

Add statements to the primary stack that enable sysplex support.

Primary stack sysplex statements are in Example 2-48.

Example 2-48 Sysplex enablement for primary stack

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

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

The VIPADYNAMIC statements enable the stack to distribute TN3270E server connections to the two servers.

VIPADYNAMIC statements added to the primary stack are in Example 2-49.

Example 2-49 VIPADYNAMIC statements for primary stack

```
VIPADYNAMIC

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.25   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25    PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21  ;TN3270 General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.1.8.21   PORT 992
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22  ;TN3270 Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.1.8.22   PORT 992
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23  ;TN3270 Payrol
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 992
                  DESTIP 10.1.7.11
                         10.1.7.21
```

```
  ;-----------------------------------------------------------------
  ;      Distribute to 10.1.7.11 via normal XCF   (no viparoute)  -
  ;      Distribute to 10.1.7.21 via IP routing   (   viparoute)  -
  ;-----------------------------------------------------------------
 ;;VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa
   VIPAROUTE      DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa

 ENDVIPADYNAMIC
```

Enable the backup stack to support sysplex functions

Add statements to the backup stack that enable sysplex support.

The statements we added to the backup stack are shown in Example 2-50.

Example 2-50 Sysplex enablement for backup stack

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

Add a VIPADYNAMIC block to the backup TCP/IP stack

The VIPADYNAMIC statements enable the stack to take over distribution when the primary stack relinquishes that responsibility. That can happen if the stack fails or upon operator command.

The statements we added are shown in Example 2-51 on page 58.

Example 2-51 VIPADYNAMIC statements for backup stack

```
VIPADYNAMIC
  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.25   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25    PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.21  ;TN3270 General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.1.8.21   PORT 992
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-----------------------------------------------------------------
  ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
  ;-----------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.22  ;TN3270 Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
```

```
                       10.1.8.22   PORT 992
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
 ;-------------------------------------------------------------------
  VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.23  ;TN3270 Payrol
  VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                       10.1.8.23   PORT 992
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;      Distribute to 10.1.7.11 via IP routing   (   viparoute)  -
 ;      Distribute to 10.1.7.21 via normal XCF   (no viparoute)  -
 ;-------------------------------------------------------------------
  VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa
;;VIPAROUTE      DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa

ENDVIPADYNAMIC
```

Customize an OMPROUTE started task to support the second stack

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

See Appendix D, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 343, for the started task procedures used for this scenario.

Customize an OMPROUTE configuration for the second OMPROUTE

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

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Additional statements must be added to support the Dynamic XCF interfaces and the Dynamic VIPA interfaces.

Example 2-52 shows the additional OMPROUTE statements for the primary stack.

Example 2-52 OMPROUTE additional statements to support D-XCF and D-VIPA: for primary stack

```
;
Global_Options Ignore_Undefined_Interfaces=yes;
;
; ************************************************************
; *Dynamic VIPA requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
```

```
; *      however this means that any address in the 10.1.8.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.8.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.1.8.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.2
     Router_Priority=0;
; ************************************************************
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to just code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
INTERFACE
     IP_Address=10.1.7.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
```

Example 2-53 shows the additional OMPROUTE statements for the backup stack.

Example 2-53 OMPROUTE additional statements to support D-XCF and D-VIPA: for backup stack

```
;
Global_Options Ignore_Undefined_Interfaces=yes;
;
; ************************************************************
; *Dynamic VIPA Range requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.8.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.8.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.1.8.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
```

```
     Attaches_To_Area=0.0.0.0
     Router_Priority=0;
; ************************************************************
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to just code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
INTERFACE
     IP_Address=10.1.7.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
```

See Appendix D, “Configuration files: Multiple TN3270E Telnet servers and SD scenario” on page 343, for the basic OMPROUTE server profiles used for this scenario.

2.3.3 Activation and verification of multiple TN3270E servers in the sysplex

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

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

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

All of the verification tasks for a standalone TN3270E server apply. Refer to “Verification of the TN3270E server” on page 32.

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

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

Start the second (backup) stack on the second system

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

```
S TCPIP
S OMP
```

Then look for the expected initialization messages.

Start the second TN3270E server on the second system

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

```
S TN3270
```

Then look for the expected initialization messages.

Use TELNET CONN displays to show TN3270 connections

In preparation for the following displays, we have connected two clients to the Dynamic VIPA 10.1.8.21 being distributed by SC30, which is running with stack profile PROFB30C and TN3270 profile TELNB30C. The D-VIPA 10.1.8.21 represents NetView® on SC30 (SC30N). The distribution method for 10.1.8.21 port 992 is set to ROUNDROBIN for our test in order to show the result of distribution upon our two client connections. Using the TSO Telnet client on SC31, we connected to 10.1.8.21 port 922 (expecting to be mapped to NetView on SC30). Using the TSO Telnet client on SC32, we again connected to 10.1.8.21 port 992 (mapping to NetView on SC30).

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

A display of the connections on SC30 in Example 2-54 on page 62 and Example 2-56 on page 63 show CONN=91 from SC31 (10.1.1.20) mapped to NetView on SC30, with an LU name of SC30B*S*02, the *S* indicating the LU pool for secure port 992 as expected.

A display of the connections on SC31 in Example 2-55 on page 62 and Example 2-57 on page 63 show CONN=8E from SC32 (10.1.1.30) mapped to NetView on SC30, with an LU name of SC31BS01, the *S* indicating the LU pool for secure port 992 as expected. The connection summary report for SC30 is shown in Example 2-54.

Example 2-54 SC30 connection to Dynamic VIPA 10.1.8.21, summary

```
RO SC30,D,TCPIP,TN3270,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 236
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   00000091    ::FFFF:10.1.1.20..1029
                                      SC30BS02 SC30N009  TA3 SNX32704
   ----- PORT:   992  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The connection summary report for SC31 is shown in Example 2-55.

Example 2-55 SC31 connection to Dynamic VIPA 10.1.8.21, summary

```
RO SC31,D,TCPIP,TN3270,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 660
            EN                                          TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   0000008E    ::FFFF:10.1.1.30..1026
                                      SC31BS01 SC30N008  TA3 SNX32704
   ----- PORT:   992  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

Detailed information about one single connection on SC30 is shown in Example 2-56.

Example 2-56 SC30 connection to Dynamic VIPA with detailed information

```
RO SC30,D,TCPIP,TN3270,T,CONN,CONN=91,DET

EZZ6065I TELNET CONNECTION DISPLAY 252
  CONNECTED: 00:52:12  09/24/2007 STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 00000091   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.1.20..1029
    DESTIP..PORT: ::FFFF:10.1.8.21..992
    LINKNAME: VIPL0A010815
  PORT:   992 QUAL: NONE
    AFFINITY: TCPIP
    STATUS: ACTIVE  SECURE        ACCESS: NON-SECURE
  PROTOCOL: TN3270            DEVICETYPE: IBM-3278-4-E
    TYPE: TERMINAL GENERIC
    OPTIONS: -TET----   3270E FUNCTIONS: *N/A*
                       NEWENV FUNCTIONS: --
  LUNAME: SC30BS02
  APPL: SC30N009
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32704  APPL SPECIFIED: SNX32704
  MAPPING TYPE:  CONN IDENTIFIER
                     OBJECT   ITEM SPECIFIC     OPTIONS
               **N/A**
                    >*DEFLUS*                 --------
    DEFLT APPL: DG GENERALUSER
                     SC30N                    --------
    USS TABLE:  **N/A**
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------ ---  ---------  -------  ----
  *******  **TSBTQ***T  EC*  BB**D****  ***STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S-------  --F  SSH------  *--ST-  S--- *TPARMS
  *M*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* TP-CURR
    PARMSGROUP: DG GENERALUSER
  *------  -----------  ---  -B-------  ------  ---- NOSSL
  *M*****  **TSBTQ***T  ECF  SBH*D****  ***STS  SDD* <-FINAL
37 OF 37 RECORDS DISPLAYED
```

Example 2-57 shows detailed information about a single connection on SC31.

Example 2-57 SC31 connection to Dynamic VIPA with detailed information

```
RO SC31,D,TCPIP,TN3270,T,CONN,CONN=8E,DET

EZZ6065I TELNET CONNECTION DISPLAY 662
  CONNECTED: 00:51:50  09/24/2007  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 0000008E   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.1.30..1026
    DESTIP..PORT: ::FFFF:10.1.8.21..992
    LINKNAME: VIPL0A010815
  PORT:   992 QUAL: NONE
    AFFINITY: TCPIP
    STATUS: ACTIVE  SECURE        ACCESS: NON-SECURE
```

```
  PROTOCOL: TN3270            DEVICETYPE: IBM-3278-4-E
    TYPE: TERMINAL GENERIC
    OPTIONS: -TET----   3270E FUNCTIONS: *N/A*
                       NEWENV FUNCTIONS: --
  LUNAME: SC31BS01
  APPL: SC30N008
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32704  APPL SPECIFIED: SNX32704
  MAPPING TYPE:  CONN IDENTIFIER
                       OBJECT   ITEM SPECIFIC     OPTIONS
                **N/A**
                      >*DEFLUS*                  --------
    DEFLT APPL: DG GENERALUSER
                       SC30N                     --------
    USS TABLE:  **N/A**
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***T  EC*  BB**D****  ***STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S-------  --F  SSH------  *--ST-  S--- *TPARMS
  *M*****  **TSBTQ***T  ECF  SSH*D****  ***STS  SDD* TP-CURR
    PARMSGROUP: DG GENERALUSER
  *------  -----------  ---  -B-------  ------  ---- NOSSL
  *M*****  **TSBTQ***T  ECF  SBH*D****  ***STS  SDD* <-FINAL
37 OF 37 RECORDS DISPLAYED
```

Use NETSTAT VCRT to show dynamic VIPA connection routing table

VCRT displays the dynamic VIPA connection routing table information.

For each table entry that represents an established dynamic VIPA connection or an affinity created by the passive-mode FTP, the DETAIL suboption additionally displays the policy rule, action information, and routing information. For each entry that represents an affinity created by the TIMEDAFFINITY parameter on the VIPADISTRIBUTE profile statement, it displays the preceding information plus the affinity-related information.

Example 2-58 shows the connections at the time the VCRT command was issued. Notice that the distributing stack knows about all of the connections because it is managing them.

Example 2-58 NETSTAT VCRT on system SC30

```
RO SC30,D TCPIP,TCPIP,N,VCRT
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.1.8.21..992
     SOURCE:  10.1.1.20..1029
     DESTXCF: 10.1.7.11
   DEST:      10.1.8.21..992
     SOURCE:  10.1.1.30..1026
     DESTXCF: 10.1.7.21
   2 OF 2 RECORDS DISPLAYED
   END OF THE REPORT
```

Notice that the non-distributing stack shows only the connections that have been distributed to it, as seen in Example 2-59.

Example 2-59 NETSTAT VCRT on system SC31

```
RO SC31,D TCPIP,TCPIP,N,VCRT
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.1.8.21..992
     SOURCE:  10.1.1.30..1026
     DESTXCF: 10.1.7.21
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

Use NETSTAT VDPT to show dynamic VIPA destination port table

VDPT displays the dynamic VIPA destination port table information.

If the DETAIL suboption is specified, the output will contain policy action information, target responsiveness values, and a WQ value (on a separate line). If DETAIL is not specified, the output will not contain policy action information or target responsiveness and WQ values.

Example 2-60 on page 65 shows the port table entries at the time of issuing the VDPT command. SC30 is currently the distributor, so it shows the ports being distributed and whether there is a ready listener on the port.

Note: The TOTALCONN field indicates the total number of connections there have been since the distribution started for the port. It does *not* represent the *current* number of connections.

Example 2-60 NETSTAT VDPT on system SC30

```
RO SC30,D TCPIP,TCPIP,N,VDPT
   DYNAMIC VIPA DESTINATION PORT TABLE:
   DEST:        10.1.8.25..20
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.25..20
     DESTXCF:   10.1.7.21
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.25..21
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.25..21
     DESTXCF:   10.1.7.21
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.21..992
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000002  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.1.8.21..992
     DESTXCF:   10.1.7.21
     TOTALCONN: 0000000001  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.1.8.22..992
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.22..992
```

```
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
    FLG: BASEWLM
  DEST:        10.1.8.23..992
    DESTXCF:   10.1.7.11
    TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
    FLG: SERVERWLM
  DEST:        10.1.8.23..992
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
    FLG: SERVERWLM
  10 OF 10 RECORDS DISPLAYED
  END OF THE REPORT
```

SC31 is not a distributor at the moment, therefore it shows no information, as seen in Example 2-61.

Example 2-61 NETSTAT VDPT on system SC31

```
RO SC31,D TCPIP,TCPIP,N,VDPT
   DYNAMIC VIPA DESTINATION PORT TABLE:
   0 OF 0 RECORDS DISPLAYED
   END OF THE REPORT
```

Use NETSTAT VIPADCFG to show current dynamic VIPA configuration

VIPADCFG displays the current dynamic VIPA configuration information from the perspective of the stack on which the command is entered.

Examples of VIPADCFG showing configuration information follow.

The primary distributor shows VIPA DEFINE, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 2-62.

Example 2-62 NETSTAT VIPADCFG on system SC30

```
RO SC30,D TCPIP,TCPIP,N,VIPADCFG
   DYNAMIC VIPA INFORMATION:
     VIPA DEFINE:
       IPADDR/PREFIXLEN: 10.1.8.25/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.1.8.21/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.1.8.22/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.1.8.23/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
     VIPA RANGE:
       IPADDR/PREFIXLEN: 10.1.9.0/24
         MOVEABLE: NONDISR
     VIPA DISTRIBUTE:
       DEST:        10.1.8.25..20
         DESTXCF:   10.1.7.11
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.25..20
         DESTXCF:   10.1.7.21
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.25..21
         DESTXCF:   10.1.7.11
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
```

```
        DEST:        10.1.8.25..21
          DESTXCF:   10.1.7.21
            SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
        DEST:        10.1.8.21..992
          DESTXCF:   10.1.7.11
            SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
        DEST:        10.1.8.21..992
          DESTXCF:   10.1.7.21
            SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
        DEST:        10.1.8.22..992
          DESTXCF:   10.1.7.11
            SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
        DEST:        10.1.8.22..992
          DESTXCF:   10.1.7.21
            SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
        DEST:        10.1.8.23..992
          DESTXCF:   10.1.7.11
            SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
        DEST:        10.1.8.23..992
          DESTXCF:   10.1.7.21
            SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      VIPA ROUTE:
        DESTXCF:     10.1.7.21
          TARGETIP:  10.1.1.20
END OF THE REPORT
```

The backup stack shows VIPA BACKUP, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 2-63.

Example 2-63 NETSTAT VIPADCFG on system SC31

```
RO SC31,D TCPIP,TCPIP,N,VIPADCFG
   DYNAMIC VIPA INFORMATION:
     VIPA BACKUP:
       IPADDR/PREFIXLEN: 10.1.8.25
         RANK: 200  MOVEABLE:           SRVMGR:
       IPADDR/PREFIXLEN: 10.1.8.21
         RANK: 200  MOVEABLE:           SRVMGR:
       IPADDR/PREFIXLEN: 10.1.8.22
         RANK: 200  MOVEABLE:           SRVMGR:
       IPADDR/PREFIXLEN: 10.1.8.23
         RANK: 200  MOVEABLE:           SRVMGR:
     VIPA RANGE:
       IPADDR/PREFIXLEN: 10.1.9.0/24
         MOVEABLE: NONDISR
     VIPA DISTRIBUTE:
       DEST:        10.1.8.25..20
         DESTXCF:   10.1.7.11
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.25..20
         DESTXCF:   10.1.7.21
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.25..21
         DESTXCF:   10.1.7.11
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.25..21
         DESTXCF:   10.1.7.21
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.21..992
         DESTXCF:   10.1.7.11
```

```
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.1.8.21..992
        DESTXCF:   10.1.7.21
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.1.8.22..992
        DESTXCF:   10.1.7.11
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.1.8.22..992
        DESTXCF:   10.1.7.21
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.1.8.23..992
        DESTXCF:   10.1.7.11
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      DEST:        10.1.8.23..992
        DESTXCF:   10.1.7.21
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    VIPA ROUTE:
      DESTXCF:     10.1.7.11
        TARGETIP:  10.1.1.10
  END OF THE REPORT
```

Use NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE

VIPADYN displays the current dynamic VIPA and VIPAROUTE information from the perspective of the stack on which the command is entered. There are two suboptions available to filter the output:

- DVIPA - Displays the current dynamic VIPA information only
- VIPAROUTE - Displays the current VIPAROUTE information only

Example 2-64 shows SC30, Netstat VIPADYN: ftp and TN3270 DVIPA addresses.

Example 2-64 NETSTAT VIPADYN on system SC30

```
RO SC30,D TCPIP,TCPIP,N,VIPADYN
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.1.8.25/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.1.8.21/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.1.8.22/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.1.8.23/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
   VIPA ROUTE:
     DESTXCF: 10.1.7.21
       TARGETIP: 10.1.1.20
       RTSTATUS: ACTIVE
   5 OF 5 RECORDS DISPLAYED
   END OF THE REPORT
```

Example 2-65 shows SC30, Netstat VIPADYN,DVIPA: filters on DVIPA only.

Example 2-65 NETSTAT VIPADYN,DVIPA on system SC30

```
RO SC30,D TCPIP,TCPIP,N,VIPADYN,DVIPA
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.1.8.25/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
     IPADDR/PREFIXLEN: 10.1.8.21/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
```

```
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
  4 OF 4 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-66 shows SC30, Netstat VIPADYN,VIPAROUTE: filters on viparoute only.

Example 2-66 NETSTAT VIPADYN,VIPAROUTE on system SC30

```
RO SC30,D TCPIP,TCPIP,N,VIPADYN,VIPAROUTE
  VIPA ROUTE:
    DESTXCF: 10.1.7.21
      TARGETIP: 10.1.1.20
      RTSTATUS: ACTIVE
  1 OF 1 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-67 shows SC31, Netstat VIPADYN: ftp and TN3270 DVIPA addresses.

Example 2-67 NETSTAT VIPADYN on system SC31

```
RO SC31,D TCPIP,TCPIP,N,VIPADYN
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
  VIPA ROUTE:
    DESTXCF: 10.1.7.11
      TARGETIP: 10.1.1.10
      RTSTATUS: ACTIVE
  5 OF 5 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-68 shows SC31, Netstat VIPADYN,DVIPA: filters on DVIPA only.

Example 2-68 NETSTAT VIPADYN,DVIPA on system SC31

```
RO SC31,D TCPIP,TCPIP,N,VIPADYN,DVIPA
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
  4 OF 4 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 2-69 shows SC31, Netstat VIPADYN,VIPAROUTE: filters on viparoute only.

Archived

Example 2-69 NETSTAT VIPADYN,VIPAROUTE on system SC31

```
RO SC31,D TCPIP,TCPIP,N,VIPADYN,VIPAROUTE
   VIPA ROUTE:
     DESTXCF: 10.1.7.11
       TARGETIP: 10.1.1.10
       RTSTATUS: ACTIVE
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

2.4 Problem determination for the TN3270E servers

This section presents methods you can use when performing problem determination for the TN3270 server:

- Reviewing the definition statements within the profile
- Using TCP/IP and Telnet commands
- Using the MSG07 statement in the TN3270 profile
- Using SMF records to capture TN3270 connection activity
- Using trace data
- Tips for multiple TN3270E servers in a Parallel Sysplex environment

2.4.1 Reviewing the definition statements within the profile

If multiple TN3270E servers are used (for example, multiple TCP/IP stacks in a sysplex environment), ensure that each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session; either the OPEN ACB request will fail, or the cross-domain session request will fail.

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

```
EZZ0621I AUTOLOG FORCING TN3270, REASON: AUTOLOGGING SCHEDULED
IEE838I TN3270TN3270 NON-CANCELABLE - ISSUE FORCE ARM
```

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

```
PORT 23 TCP TN3270 NOAUTOLOG
```

2.4.2 Using TCP/IP and Telnet commands

A number of TCP/IP and Telnet commands can be used to assist with TN3270 server problem determination. The more popular methods are discussed in this section.

Using OBEYFILE to turn selected debug options on and off

TN3270-specific debug messages can be turned on or off to diagnose TN3270 problems. Several types of debug messages are available.

- Summary messages indicate important status changes.
- Detail messages indicate a problem was detected.
- General messages indicate an important event.
- Trace messages show data to and from the client, and to and from VTAM.
- Flow messages show entry into and exit from modules.

Message generation is controlled with the DEBUG statement. The statement can be specified in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. A connection must have the DEBUG statement mapped to it for messages to be generated. Whenever a DEBUG message is generated, a CTRACE entry is also generated.

The format of the DEBUG statement is shown in Example 2-70. The parameters can involve subparameters, which we do not show in our examples. For details of the complete syntax, see the DEBUG statement description in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For comprehensive suggestions on the use of the DEBUG statement, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Example 2-70 DEBUG statement format

```
DEBUG parameter1  parameter2
```

As shown in Table 2-2, the DEBUG statement has two parameters: the level of tracing and the destination for trace messages. The two parameters can be specified in any combination. Note that the table does *not* imply a one-to-one relationship.

Table 2-2 DEBUG statement parameters

Trace level	Trace destination
OFF	CONSOLE (default for exception)
EXCEPTION	JOBLOG (default for summary, detail, trace)
SUMMARY	CTRACE (default for flow, profile)
DETAIL	
TRACE	
FLOW	
PROFILE	

Note: The PROFILE parameter can be specified in TELNETGLOBALS only. It has no effect on the other DEBUG statements.

If DEBUG messages are being used primarily for problem diagnosis, the VARY TCPIP,,OBEYFILE command can be used to keep the number of messages low. Bring up TN3270 initially without DEBUG coded. When a problem appears, issue a VARY TCPIP,,OBEYFILE command for a TN3270 profile that includes the DEBUG statement. Only new connections to the new profile will produce messages. After data is obtained, issue another VARY TCPIP,,OBEYFILE command for a TN3270 profile that omits the DEBUG statement or specifies DEBUG OFF.

If the client identifier of the client having the problem is known (perhaps the client's current IP address), include DEBUG in a PARMSGROUP statement. Then, using PARMSMAP, map that group to the client. Debug messages for only that client will be issued. After data is obtained, issue another VARY TCPIP,,OBEYFILE command for a TN3270 profile that omits the DEBUG statement or specifies DEBUG OFF.

An example of the profile statements and obeyfile command for activating the trace just suggested is shown in Example 2-71. Assume the profile member name for turning the trace on is TELNDBON. Note that, to avoid changing anything else in the current profile (TELNB30A), TELNDBON is a copy of TELNB30A, with these three statements added.

Example 2-71 Preparing a DEBUG TRACE statement for OBEYFILE command

```
profile statements in member TELNDBON
   . . .
PARMSGROUP DEBUGIT DEBUG TRACE JOBLOG ENDPARMSGROUP
IPGROUP DEBUGIPGRP 10.1.1.20 ENDIPGROUP
PARMSMAP DEBUGIT IPGRP,DEBUGIPGRP
   . . .

VARY TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNDBON)
   EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 624
               TCPIP.TCPPARMS(TELNDBON)
   EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 625
               TCPIP.TCPPARMS(TELNDBON)
   EZZ6018I TELNET PROFILE UPDATE COMPLETE FOR PORT    23
   EZZ6038I TELNET COMMAND OBEYFILE  COMPLETE
```

When we display the CLIDs and OBJects for port 23, we see the debug options that were just set, as shown in Example 2-72.

Example 2-72 Display profile showing debug options ON

```
D TCPIP,TN3270,T,CLID,PORT=23,PROFILE=ALL,DET,MAX=*
   EZZ6081I TELNET CLIENTID DISPLAY 689
   CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
   NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
   ------------------ ------ --------- -------- ---------- --------
   IPGRP
     DEBUGIPGRP
                          0 PARMSGRP  DEBUGIT             --------
   NULL
     NULL
                          0 DEFAPPL   TSO                 --------
   -------------------------------------- PROF: CURR CONNS:      0
   NULL
     NULL
                          1 DEFAPPL   TSO                 --------
   ----- PORT:    23  ACTIVE              PROF: 0001 CONNS:      1
   ------------------------------------------------------------
   12 OF 12 RECORDS DISPLAYED

D TCPIP,TN3270,T,OBJ,PORT=23,PROFILE=ALL,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 691
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   ARAPPL
    SC30N*          0
                                                             -A------
    NVAS*           0
                                                             -A--Q---
    TSO*            0
                                                             -A-D----
    *               0
                                                             -A------
    *DEFAPPL        0
                                                             -A------
   DEFAPPL
    TSO             0 NULL      NULL
                                                             --------
   LUGRP
```

```
    *DEFLUS*        0
                                                          --------
   PARMSGRP
    DEBUGIT         0 IPGRP     DEBUGIPGRP
                                                          --------
    *DEFAULT          -------NO MAPPING---------
                                                          --------
    *TGLOBAL          -------NO MAPPING---------
                                                          --------
    *TPARMS           -------NO MAPPING---------
                                                          --------
   ---------------------------------------- PROF: CURR CONNS:      0
```

The profile statements and obeyfile command for deactivating the trace are shown in Example 2-73. Assume the profile member name for turning the trace off is TELNDBOF.

Example 2-73 Preparing a DEBUG EXCEPTION statement for OBEYFILE command

```
profile statements in member TELNDBOF
   . . .
PARMSGROUP DEBUGIT DEBUG EXCEPTION ENDPARMSGROUP
IPGROUP DEBUGIPGRP 10.1.1.20 ENDIPGROUP
PARMSMAP DEBUGIT IPGRP,DEBUGIPGRP
   . . .

VARY TCPIP,TN3270,OBEYFILE,TCPIP.TCPPARMS(TELNDBOF)
   EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 861
              TCPIP.TCPPARMS(TELNDBOF)
   EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 862
              TCPIP.TCPPARMS(TELNDBOF)
   EZZ6018I TELNET PROFILE UPDATE COMPLETE FOR PORT    23
   EZZ6038I TELNET COMMAND OBEYFILE  COMPLETE
```

To avoid changing anything in the current profile (TELNDBON), TELNDBOF must be a copy of TELNDBON with these three statements replacing their equivalent statements in TELNDBON. (Both TELNDBON and TELNDBOF are based on the contents of TELNB30A).

Using VARY DEBUG OFF to turn off all debug options

As an alternative to using the OBEYFILE command and the DEBUF OFF statement, use the VARY TCPIP,,TELNET,DEBUG,OFF command. Using OBEYFILE only turns off debug for new connections. Any connections on the old profile continue to produce debug messages.

The VARY command can be issued to turn off DEBUG for *all* connections associated with *all* profiles, including the current profile. Summary messages for CONN DROP due to errors or time-outs will also be suppressed.

To turn on DEBUG again, issue a VARY TCPIP,,OBEYFILE command with the debug option specified in the TN3270 profile. Use DEBUG EXCEPTION to retain these messages.

The DEBUG OFF command is shown in Example 2-74.

Example 2-74 DEBUG OFF command used to turn off all debug options

```
V TCPIP,TN3270,T,DEBUG,OFF
   EZZ6038I TELNET COMMAND DEBUG OFF COMPLETE
```

The options are listed again, after the DEBUG OFF command, as shown in Example 2-75.

Example 2-75 Display PROFILE to see DEBUG settings after DEBUG OFF command

```
D TCPIP,TN3270,T,PROF,PORT=23,DET
      .........
   DIAGNOSTICS
      DEBUG              VARYOFF
      DEBUG ROUTING      CTRACE
      FULLDATATRACE
      .........
```

Using VARY ABENDTRAP to set abend traps

The **VARY TCPIP,,TELNET,ABENDTRAP,*module,rcode,instance*** command sets an abend trap based on unique Telnet return codes set in Telnet modules. You will normally use this command at the direction of IBM Support Center personnel. Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details related to the use of this command.

Using NETSTAT CONN with APPLDATA

You can optionally display additional application connection data using the APPLDATA parameter on the NETSTAT CONN and NETSTAT ALLCONN commands. Example 2-76 contrasts the output of two NETSTAT CONN commands: one without the APPLDATA parameter 1, the other with the APPLDATA parameter 2.

The TN3270 server populates the APPLDATA field with connection data that is documented in Appendix “Application Data” found in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. The TN3270 appldata fields shown for the connection are the component ID, LU name, the SNA application name, connection mode, client type, security method, security level, and security cipher 3.

Example 2-76 NETSTAT CONN without and with the APPLDATA option

```
D TCPIP,TCPIP,N,conn                                  1
USER ID  CONN     STATE
TN3270B  00000111 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.20..23
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1028

D TCPIP,TCPIP,N,conn,appldata                         2
USER ID  CONN     STATE
TN3270B  00000111 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.20..23
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1028
  APPLICATION DATA:  EZBTNSRV SC31BB05 SC31TS03 3T B    3
```

You can filter the output of the NETSTAT CONN,APPLDATA command by adding the APPLD filter option and specifying the filter criteria. The APPLDATA field is a total of 40 bytes. By using an asterisk (*) in the filter criteria, you can wildcard on any part of the 40 bytes. Example 2-77 shows a few filter criteria strings being used.

Example 2-77 NETSTAT CONN APPLDATA with APPLD filter

```
D TCPIP,TCPIP,N,conn,appldata,appld=*tnsrv*
USER ID  CONN     STATE
TN3270B  00000111 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.20..23
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1028
  APPLICATION DATA:  EZBTNSRV SC31BB05 SC31TS03 3T B
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

```
D TCPIP,TCPIP,N,conn,appldata,appld=*sc31*
USER ID  CONN     STATE
TN3270B  00000111 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.20..23
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1028
  APPLICATION DATA:  EZBTNSRV SC31BB05 SC31TS03 3T B
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

2.4.3 Using the MSG07 statement in the TN3270 profile

The MSG07 parameter is very helpful when diagnosing problems. It allows TN3270 to send a message to the client indicating an error occurred and what the error was. Something simple like a mistyped application name can be corrected by the end user without additional help.

Even for more difficult problems, MSG07 provides a good starting point. We recommend that MSG07 always be coded in order to send a notification to the user of connection failure issues, unless there are reasons not to send error messages to the client.

Use the MSG07 parameter statement to activate logon error message processing. Specifying this statement provides information to the client when a session attempts to the target application fails. If NOMSG07 is specified, the connection is dropped if a session initiation error occurs.

The MSG07 and NOMSG07 statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. They cannot be coded in the BEGINVTAM block.

MSG07 being used in the TELNETPARMS block is shown in Example 2-78.

Example 2-78 MSG07 used in TELNETPARMS block

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

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

2.4.4 Using SMF records to capture TN3270 connection activity

SMF records are written when an end user establishes a session (SMF LOGN or Telnet SNA Session Initiation record, subtype 20) and when the session is ended (SMF LOGF or Telnet SNA Session Termination record, subtype 21). Optional SMF recording is controlled by using the SMFINIT and SMFTERM statements.

SMFINIT and SMFTERM can be coded in TELNETGLOBALS, TELNETPARMS, and PARMSGROUP. An example of SMFINIT/SMFTERM being used in the TELNETPARMS block is shown in Example 2-79.

Example 2-79 turns off all 118 format records and uses standard 119 format records.

Example 2-79 SMFINIT/SMFTERM used in TELNETPARMS block

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

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

2.4.5 Using trace data

Using a packet trace to capture data buffers

Use the VARY TCPIP,,PKTTRACE command to trace data buffers between client and server. The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace. You will normally use this command at the direction of IBM Support Center personnel. Refer to the following sources for details:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535
- *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis

Using a data trace to capture socket data

Use the VARY TCPIP,,DATTRACE command to trace socket data (transforms) into and out of the physical file structure (PFS). The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace. You will normally use this command at the direction of IBM Support Center personnel. Refer to the following sources for details:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535
- *z/OS MVS IPCS Commands*, SA22-7594, for formatting and analysis

Using the CTRACE system component to trace TN3270 processes

If a problem is not resolved using the preceding tools, IBM service will likely need a CTRACE with option Telnet. CTRACE, with only the Telnet option, gives very complete information about the TN3270 processes. To debug almost any TN3270 problem, no other CTRACE option is needed. TN3270 running as its own procedure continues to use the SYSTCPIP component trace, but has fewer trace options to be specified.

A component trace SYS1.PARMLIB member is supplied for TN3270. The member name is CTIEZBTN. Trace setup for TN3270 running as its own procedure is the same as for the TCP/IP stack, except for having fewer trace options available. For a complete discussion of the CTRACE options and trace setup, refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

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

2.4.6 Tips for multiple TN3270E servers in a Parallel Sysplex environment

If multiple TN3270E servers are used (for example, multiple TCP/IP stacks in a sysplex environment), ensure that each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session; either the OPEN ACB request will fail, or the cross-domain session request will fail.

2.5 Additional information sources for the TN3270E server

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- TN3270 is defined by RFC 1646
- TN3270E is defined by RFC 1647 and RFC 2355
- *3270 Data Stream Programmer's Reference*, GA23-0059
- *z/OS Communications Server: IP and SNA Codes*, SC31-8791
- *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535

Archived

Archived

3

File Transfer Protocol

File Transfer Protocol (FTP) allows you to move files between any computers that support the TCP/IP-standard FTP protocols—mainframes, midrange systems, PC servers, and desktop systems. FTP is the name of the application and the protocol that allows you to transfer files in a TCP/IP network. This chapter focuses on the FTP functions that are available in the z/OS Communications Server. The FTP protocol uses a client/server model and the z/OS Communications Server ships with both an FTP server (FTPD) and an FTP client.

This chapter discusses the following FTP topics.

Section	Topic
3.1, “Conceptual overview of FTP” on page 80	The basic concepts of FTP.
3.2, “Basic FTP without security” on page 84	Key characteristics of basic FTP and how to configure and verify a single server implementation.
3.3, “Multiple FTP servers in a sysplex” on page 108	How to implement multiple FTP servers in a sysplex, one server per system image.
3.4, “FTP client via batch” on page 119	How to run an FTP client in batch.
3.5, “Miscellaneous configurations of FTP” on page 122	Miscellaneous FTP functions not covered in detail in this book: REXX™ API, generic FTP server in a CINET environment, the network management interface with SMF.
3.6, “Problem determination for FTP” on page 124	Problem determination techniques for FTP.
3.7, “Additional information sources for FTP” on page 124	References to additional reading sources for FTP.

© Copyright IBM Corp. 2008. All rights reserved.

3.1 Conceptual overview of FTP

As illustrated in Figure 3-1, FTP is one of the standard applications provided with the z/OS Communications Server. It executes under the z/OS UNIX environment, uses the LE C-sockets API, and passes data through the UNIX Logical and Physical File Systems.

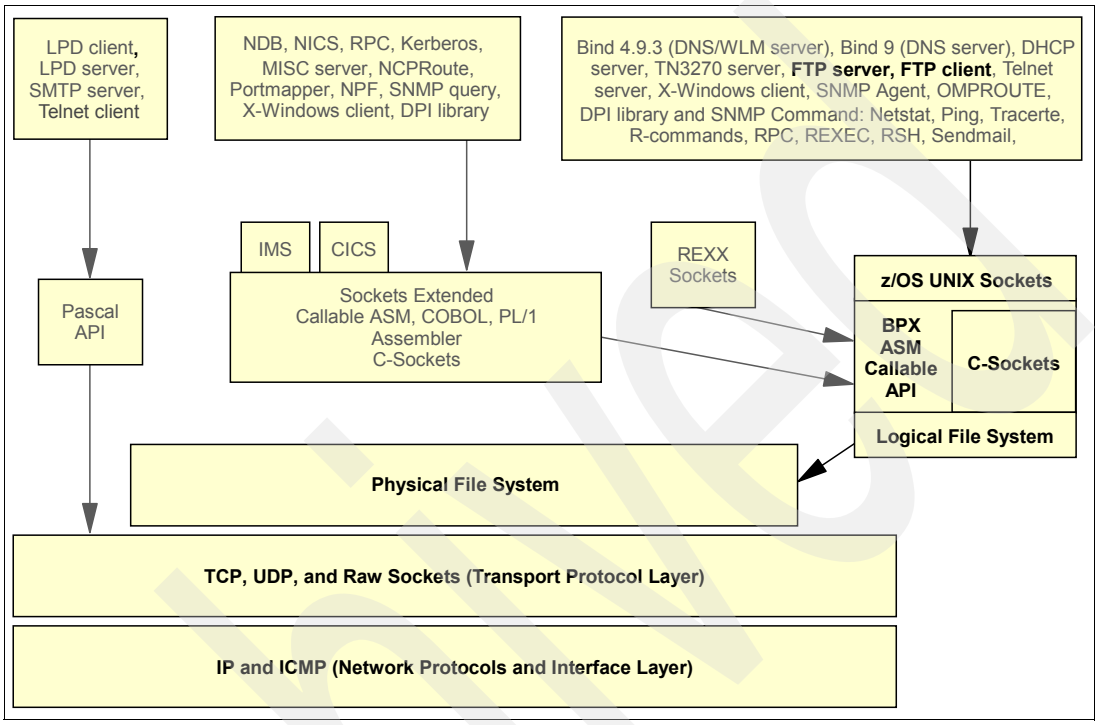


Figure 3-1 z/OS FTP client/server application services

This section includes the following topics:

- What is FTP?
- How does FTP work?
- How can FTP be used?

3.1.1 What is FTP?

FTP provides a fast and reliable method to transfer files in a TCP/IP network. FTP also allows for communication between platforms that have different character encodings and file structures by hiding such details from the user. With a few simple FTP commands, one can easily transfer a file from one platform to another regardless of whether the two platforms are similar.

There are two devices involved in an FTP session: the local host is the client, and a remote host is the server. Using the FTP command and subcommands, you can sequentially access multiple hosts without leaving the FTP environment. The local host or FTP client is the TCP/IP host that initiates the FTP session. The FTP server is the TCP/IP host to which the client's session is established. The server provides the responses to the client's commands and subcommands. z/OS Communications Server FTP includes translation facilities for ASCII/EBCDIC translation to support host file transfer to and from a variety of host platforms and operating systems.

3.1.2 How does FTP work?

An FTP session is initiated when an FTP client connects to an FTP server using the TCP protocol. The FTP protocol requires the FTP server to use two TCP ports. One TCP port is the control connection over which information such as user ID and password is transmitted. All FTP commands, subcommands, and responses are exchanged over this connection. Well-known port 21 is used as the default for the control connection port on the FTP server. The other TCP port is the data connection, which is used for transferring the contents of files based on the FTP client's requests. The output of the **`ls`** or **`dir`** FTP subcommands is also sent over the data connection. Well-known port 20 is the default for the data connection port on the FTP server. Refer to *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, for details about FTP usage, commands, and subcommands.

During an FTP session it is important to keep track of which machine is the client and which is the server, because this determines whether you use a **`get`** command or a **`put`** command to move files. The **`get`** command is always used to copy files from the server to the client and the **`put`** command is used to copy files from the client to the server.

Configuration files

The FTP server and client can each have its own optional FTP.DATA configuration data set. The z/OS Communications Server provides a sample FTP.DATA for the FTP server in SEZAINST(FTPSDATA) and a sample for the FTP client in SEZAINST(FTCDATA). *z/OS Communications Server: IP Configuration Reference*, SC31-8776, covers the configuration statements in detail and indicates which statements are appropriate for the server or the client.

FTP server job name

The FTP server forks once at startup, and forks again each time a new connection is accepted. The job name of the initial address space is based on the started procedure. The server's listener thread (first forked thread) always adopts the started task's name (proc name) suffixed with the number 1. For example, if the proc name is FTPD, then the listener's name is always FTPD1. For all subsequent forks, a new job name is chosen in one of three ways:

- If the proc name is already eight characters long, then the job name remains that same eight character name, and does not change across forks.
- Assigned by _BPX_JOBNAME environment variable
- Assigned by z/OS UNIX

_BPX_JOBNAME

If the original job name is less than eight characters, and *_BPX_JOBNAME* is made available to the FTPD server at start up, all forked threads, after the listener's, result in the jobname specified by *_BPX_JOBNAME*.

Assigned by z/OS UNIX

If the original job name is less than eight characters, and *_BPX_JOBNAME* is not specified, then z/OS UNIX assigns forked job names by appending a number to the job name. For example, if a started procedure named FTPD is used to start the FTP server, then the job name of the listener is FTPD1. For each subsequent fork (which is created in behalf of a new client connection), the job name becomes FTPD*x,* where *x* is a number between two and nine (FTPD2 - FTPD9).

z/OS UNIX uses RACF's *BPX.DAEMON* FACILITY class profile to manage the UNIX security context of threads that are forked by certain servers. These servers change the security environment of the process in which they execute in order that the client threads execute

under the security permissions set up for the client's user ID. For example, the FTPD daemon creates a new z/OS UNIX process for every FTP client connecting to it. After the new process is created, the daemon changes the security environment of the process so that it is associated with the security context of the logged-in user, instead of the server's user ID. The RACF FACILITY class resource *BPX.DAEMON* is used by the servers for changing the security context during this process.

3.1.3 How can FTP be used?

There is an FTP client program and a separate FTP server program provided with z/OS Communications Sever.

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

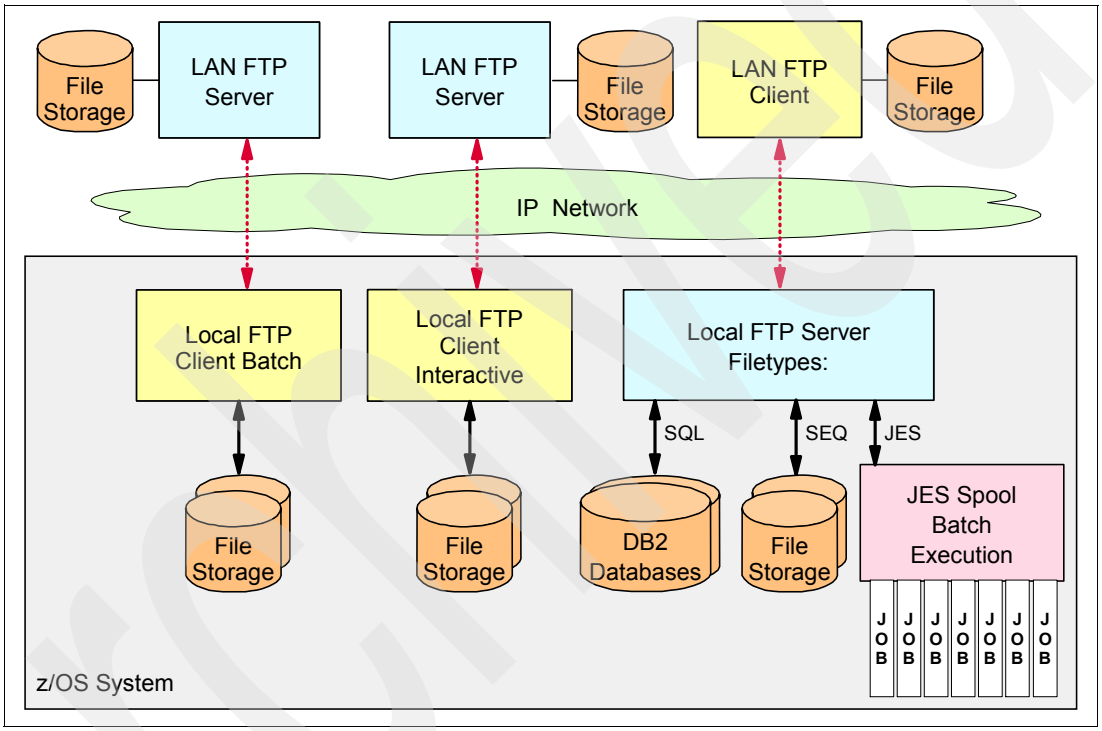


Figure 3-2 FTP client/server relationships

Features of FTP

FTP in the z/OS Communications Server includes the following features:

- Access to traditional MVS data sets (including PDS and PDS/E data sets).
- Access to files in the UNIX System Services file system.
- Support for MVS batch jobs using the JES file type.
- Support for SQL queries using the SQL file type.
- Translation tables supporting many different languages, including both single-byte character sets (SBCS), double-byte character sets (DBCS), and unicode character set 2 (UCS-2).
- Secure communications using Transport Layer Security (TLS) or Kerberos.
- Secure communications using Application Transparent TLS (AT-TLS)
- Access control via support for user exits.

- Support for anonymous login.
- A customizable welcome banner, login message, and directory readme files.
- Access to traditional MVS data sets (including PDS and PDS/E data sets).
- The client can be invoked from TSO, from the UNIX system services shell, as a batch job, or by an application programming interface (API).
- Support for FTP proxy connections via a SOCKS server.
- Controls in FTP.DATA for server and client operations.
- Support for UNICODE TP reporting of UTF-8, UTF-16, UTF-16LE, and UTF-16BE encoding.

The client and the server can both be executed with or without encryption security. Both can support SOCKS proxy protocols to accommodate file transfers within a firewall environment. A single generic FTP server can be implemented on a single LPAR that has multiple stacks. The single server can listen on each stack. Load balancing and availability requirements across FTP servers within a sysplex can be implemented. Multiple FTP server instances can be implemented in a multiple stack (CINET) environment. The FTP client can run in batch, in a TSO environment, or under program control using one of the available API interfaces.

For users interested in setting up basic FTP without security in order to get a simple FTP server up and running, we recommend a configuration similar to the one discussed in 3.2, “Basic FTP without security” on page 84. You can then later add security definitions to the basic configuration.

For users interested in secure FTP, refer to using the native TLS configuration or using a stack-wide security solution such as Application Transparent Transport Layer Security (AT-TLS) or IPSec. In addition to native TLS support, the FTP server and client can use AT-TLS to manage TLS security. TLS managed by AT-TLS (TLSMECHANISM ATTLS) supports more security functions than TLS managed natively by the FTP (TLSMECHANISM FTP). Be aware of these AT-TLS capabilities and requirements when planning the preferred AT-TLS support for FTP:

- Specify the label of the certificate to be used for authentication instead of using the default
- Support SSL session key refresh
- Support SSL sysplex session ID caching
- Trace decrypted SSL data for FTP in a data trace
- Receive more detailed diagnostic messages in syslogd
- There are no restrictions for the FTP ATTLS function.
- Policy Agent must be active for FTP ATTLS to work.
- TLS security defined natively to FTP will continue to be available in addition to AT-TLS.

All TLS and AT-TLS security related scenarios are discussed in *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535).

For users interested in load balancing FTP across multiple LPARs we recommend the high availability scenarios, as discussed in *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534.

The following scenarios are discussed in this book:

- Basic FTP without security
- Multiple FTP servers in a sysplex
- FTP client via batch
- Miscellaneous configurations of FTP

3.2 Basic FTP without security

In this section we discuss FTP configured with common configuration options but without any data security beyond user IDs and passwords. In practical terms, we discuss the transmission of FTP data without any encryption. This is the easiest and quickest way to get a basic FTP server configuration implemented.

The following topics are discussed:

- Description of basic FTP without security
- Planning for the basic FTP environment without security
- Configuration of basic FTP without security
- Activation and verification for basic FTP without security

3.2.1 Description of basic FTP without security

Dependencies of basic FTP

Implementing FTP without security requires minor changes to the TCP/IP configuration, some definitions in the SERVAUTH SAF class, and an FTP.DATA data set. Starting syslogd prior to starting FTP is highly recommended for retaining messages from the FTP server.

Advantages of basic FTP

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

Considerations for basic FTP

Without using z/OS Communications Server security capabilities, you may need to implement necessary security elsewhere (in your firewalls or routers, for example) or you may elect to implement some level of IPSec.

3.2.2 Planning for the basic FTP environment without security

Each operating system has unique requirements for allocating files or data sets in its file system. These requirements differ so widely between operating systems that it has been impossible to develop a single protocol that embraces all requirements for all operating systems. In order to cover all requirements, the FTP protocol implements a SITE command which enables an FTP client to send site-specific parameters to the FTP server over the control connection.

Option to change server reply code from 250 to 226

This option can improve information management and log analysis. It enables you to configure FTP server to reply with 226 instead of 250 after a successful file transfer. It provides more flexibility when obtaining access through firewalls.

Generally, reply code 226 or 250 is used after a successful file transfer, after LIST commands, and after NLST commands. Reply code 250 (not 226) is used for a broader class of FTP commands, such as RNTO, DELE, MKD, RMD, CWD.

The FTP server's current implementation sends a 250 reply, even though it has already closed the data connection. This should be changed to 226 in order to comply with the RFC standard. This change is being implemented via an FTP.DATA parameter to allow local control in case there is a local dependency on the 250 reply.

Use the **REPLY226** statement to direct the FTP server to reply to the FTP client with reply code 226 instead of reply code 250 to command sequences described in RFC-959 that allow the server to choose between reply 226 and reply code 250.

The options for this statement are:

- **FALSE**: directs the server to reply to the client with code 250 after successful file transfer, and after other FTP commands where the server is allowed to choose between reply code 250 and reply code 226. This is the default.
- **TRUE**: directs the server to reply to the client with reply 226 instead of reply code 250 after successful file transfer, and after other FTP commands where the server is allowed to choose between reply code 250 and reply code 226.

Restriction: A server is not always permitted to select reply 226 instead of reply 250. The REPLY226 setting does not override RFC-959 in these cases.

For example, RFC-959 stipulates the server must reply with reply 250 to RMD (remove directory); the REPLY226 setting does not affect the reply code for RMD commands.

Parameter for FTP client to specify FTP.DATA

An option at the client invocation level enables the FTP client to specify a dataset, UNIX file, or ddname to override the FTP.DATA search order. This option is implemented via an FTP.DATA parameter to allow customer control over its usage.

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

Table 3-1 Option to set the FTP.DATA file through the -f parameter

TSO environment	z/OS UNIX System Service shell
0. -f parameter 1. SYSFTPD DD statement 2. tso_prefix.FTP.DATA 3. userid.FTP.DATA 4. /etc/ftp.data 5. SYS1.TCPPARMS(FTPDATA) dataset 6. tcpip_hlq.FTP.DATA	0. -f parameter 1. $HOME/ftp.data 2. userid.FTP.DATA 3. /etc/ftp.data 4. SYS1.TCPPARMS(FTPDATA) 5. tcpip_hlq.FTP.DATA

Note: It is important to specify an existing ftp.data file with the -f parameter. If the specified file is not found, then the standard ftp.data search order is performed. No error message is issued.

Platform-specific features via SITE/LOCSITE commands

When a user on your z/OS system starts the FTP client, a set of default local SITE parameters is in effect. The default values can be specified in the FTP.DATA data set. If an FTP.DATA data set cannot be found, a set of hard coded values is used. The user can change these parameters during the FTP session by using the LOCSITE command.

For details about the SITE, LOCSITE, and FTP.DATA client statement parameters, consult *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Data set attributes

When an FTP client issues a `put` to transfer a file to the z/OS FTP server, the FTP server needs specific parameters in order to allocate a data set. These parameters include record

format (RECFM), record length (LRECL), unit type (UNIT), and blocksize (BLKSIZE), plus many others, depending on the specific operation requested. The FTP server has a set of default values for all the parameters it may need. The client can change many of these values for the current FTP session via the SITE command.

If you use the z/OS FTP client function and you retrieve a file from an FTP server somewhere in your IP network, the FTP client also needs a set of parameters similar to those of the z/OS FTP server to allocate a data set in MVS. Again, a set of default values exists for the z/OS FTP client, but a user may change these via a LOCSITE command.

You do not necessarily need to specify all the allocation attributes of an MVS data set; you may instead use the Storage Management System (SMS) of IBM Data Facility Systems Managed Storage. You have in both the SITE and the LOCSITE command an option to specify values for the three main SMS parameters: dataclass, mgmtclass, and storclass. These SMS options are:

- Data class (site/locsite dataclass=), which is a collection of data set allocation attributes, for example, space requirements, record format, data set type, or retention period.
- Management class (site mgmtclass=), which is a collection of management attributes, for example, migration rules, backup frequency, or rules for release of unused space.
- Storage class (site storclass=), which is a collection of service attributes, for example, availability requirements and requested storage subsystem response time.

Consult your storage administrator for a list of available SMS parameters in your installation.

Directory mode or data set mode

The directory mode or data set mode specifies how the output from a directory command submitted to the z/OS FTP server should look. Working with FTP employs the notion of a directory and a hierarchy of directories. When MVS is the FTP server, the client still uses the directory notion and the server must transform this notion into the traditional MVS file system structure. The client can switch between the two modes by using the SITE command specifying either DIRECTORYMODE or DATASETMODE.

DATASETMODE is the normal MVS way of displaying MVS data set names. An example of this output is shown in Example 3-1.

Example 3-1 Output of dir command in DATASETMODE

```
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit  Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
WTLSTG 3380K  05/13/96  1   10  FB      80  6160  PO  DB2.CNTL
WTLSTG 3380K  05/13/96  1    6  VB    4092  4096  PS  DB2.OUTPUT
WTLSTG 3380K  05/13/96  1    2  FB      80  3120  PO  ESA4.ISPPROF
WTLSTG 3380K  05/13/96  1    9  FB      80  6160  PO  ISPF.CLIST
WTLSTG 3380K  05/13/96  1   10  FB      80  6160  PO  ISPF.ISPPLIB
WTLSTG 3380K  05/13/96  1    1  FB      80  6160  PO  ISPF.TEST.CLIST
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  ISPF.TEST.WORKDSN
WTLSTG 3380K  05/13/96  1    2  FB      80  3120  PO  ISPFESA.ISPPROF
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  PRINT
WTLSTG 3380K  05/13/96  1    8  VA     125   129  PS  SPFLOG1.LIST
WTLSTG 3380K  05/13/96  1    1  FB      80   800  PS  SPFTEMP1.CNTL
250 List completed successfully.
808 bytes received in 1.3 seconds (0 Kbytes/s)
```

However, we might want output that more closely resembles the UNIX style. DIRECTORYMODE uses the value of your TSOPREFIX setting in RACF as your default

directory. If you do not maintain TSO logon information in RACF, your user ID will be used as your default directory.

Each qualifier level in the data set name is considered a directory. A directory may contain data sets or subdirectories. A partitioned data set is considered a directory, and the individual members as files in that directory. You may step down the hierarchy by using **CD** commands to name the next low-level qualifier you want to view. You may step up to the root by using **CD ..** commands.

An example of DIRECTORYMODE is shown in Example 3-2.

Example 3-2 Output of dir command in DIRECTORYMODE

```
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit  Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
Pseudo Directory                                       DB2
Pseudo Directory                                       ESA4
Pseudo Directory                                       ISPF
Pseudo Directory                                       ISPFESA
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  PRINT
Pseudo Directory                                       SPFLOG1
Pseudo Directory                                       SPFTEMP1
250 List completed successfully.
493 bytes received in 0.84 seconds (0 Kbytes/s)
```

Data type, structure, and mode

At first glance it may seem to be a trivial matter to transfer files between different computer systems, but when you take a closer look you soon discover a range of issues created by the diversity of computer architectures in a typical IP network. Some operating systems use 7-bit ASCII to represent character data. Others use 8-bit ASCII or EBCDIC, just to mention the most obvious. Some operating systems organize files into records, while others treat files as continuous streams of data, possibly without any encoded notion of record boundaries. (In this case, it is up to the program reading or writing the data to impose a structure onto the data stream.)

The FTP protocol can help deal with these issues, but you must select the proper options in order to let FTP transfer a file in such a way that it is usable on the receiving system.

FTP always transfers data in 8-bit bytes, the *transfer size*. If the sending or receiving system uses another byte length, it is up to the FTP client and the FTP server to implement the proper conversion between local byte sizes and the FTP transfer size. When FTP transfers ASCII data, it always transfers it in 8-bit bytes, where the bits are used to encode the ASCII character according to a specific ASCII standard, which is called NVT-ASCII (Network Virtual Terminal ASCII as defined in the TELNET protocol). This implies that when you transfer ASCII type data between two ASCII hosts, a translation from the local ASCII representation to NVT-ASCII for transmission and back to the receiving hosts local ASCII representation always takes place.

When MVS is involved in an ASCII type transfer MVS translates the received NVT-ASCII into EBCDIC and translates data to be transmitted from EBCDIC into NVT-ASCII.

When you request an FTP file transfer, you can characterize the transfer by means of three attributes: type, structure, and mode.

Archived

Data type

Data type, also known as transfer type or representation type, indicates how the bits of data should be interpreted by the receiver. There are three values: ASCII, EBCDIC, and IMAGE.

ASCII When you set the data type to ASCII, the receiver knows that the data is character data and that each line of data is terminated via a control sequence of Carriage Control plus Line Feed (CRLF), which in ASCII is X'0D0A'. If MVS is the receiving side, data is translated from NVT-ASCII to EBCDIC and the CRLF sequences are removed from the data stream and substituted by traditional MVS record boundaries according to the current settings of the SITE/LOCSITE parameters: RECFM and LRECL. If RECFM is fixed, the data records are padded with extra spaces in order to fill up a record. If MVS is the sending side, the data is translated from EBCDIC into NVT-ASCII and, based on existing record boundaries, CRLF sequences are constructed and inserted into the ASCII data stream. A data type of ASCII is the default data type in all FTP implementations.

EBCDIC A data type of EBCDIC means that the data transferred is EBCDIC data. In such a case, no translation to NVT-ASCII or from NVT-ASCII takes place in MVS. The 8-bit EBCDIC bytes are transferred as they are. If you transfer text data between two EBCDIC systems, a data type of EBCDIC is the most efficient way to transfer the data. Most ASCII hosts will reject a data transfer where you specify a data type of EBCDIC. Some will treat it as an ASCII transfer, but the point where the translation takes place is at their end of the FTP connection, and not in MVS.

IMAGE A data type of IMAGE means that the data will be transmitted as contiguous bits packed into the 8-bit FTP transfer byte size. No translation takes place, neither at the sending nor at the receiving side. You normally use this data type for binary data, such as program files. If you transfer text data between two similar ASCII hosts, it will often be more efficient to use an IMAGE data type instead of an ASCII data type. As the two systems use exactly the same ASCII representation, there is no need to impose the overhead of translating to and from NVT-ASCII.

Both ASCII and EBCDIC data types have a second attribute, *format control*.

Non-print The text data does not include any vertical format control characters. This format control is the only one you will find in the MVS FTP implementation. When you set data type to ASCII, the format control defaults to non-print.

TELNET The text data includes TELNET control characters. This is not commonly used.

CC The text data includes Carriage Control (ASA) control characters, according to the FORTRAN implementation.

Data structure

Structure refers to how the data is stored by the receiver. The three possible values are file, record, and page:

File The file has no internal structure and is considered to be a continuous sequence of bytes. File structure can be used with all transfer modes and data types, and is the most widely implemented.

Record The file is made up of sequential records. This is a relatively simple matter to deal with as long as we talk about text files. The ASCII data type with CRLF sequences can be seen as a case of record data. All data types are generally supported for record structure. In CS for z/OS IP, the explicit use of record structure is only supported with stream mode transfers. When record structure is explicitly used, each record is terminated by an end-of-record (EOR)

sequence, which is X'FF01'. End-Of-File (EOF) is indicated by X'FF02'. If the data contains X'FF' bytes, each X'FF' byte is transmitted as two bytes, X'FFFF', in order to preserve the original value. In CS for z/OS IP both the FTP server and the FTP client can support this record structure.

Page A third structure value is page structure. It is not used in conjunction with MVS, and CS for z/OS IP does not support it, either in the FTP client or in the FTP server.

Transfer mode

Transfer mode refers to the organization of the data as it is transmitted. The three possible values are stream, block, and compress.

Stream Data is transmitted as a stream of bytes. The data is passed with little or no extra processing. With stream mode, the data type is used to determine if any processing at all should be applied to the data, such as translation of text data or CRLF processing. There is no restriction on data type or data structure. If record structure is used, the end of file is indicated via the EOF control sequence (X'FF02'). If file structure is used, the end of the file is indicated when the sending host closes the data connection. Stream mode is the default transfer mode, and the most commonly used.

Block In block mode data is sent as a series of data blocks. Each block is preceded by a header. The header contains a count field of the number of bytes in the block (excluding the header itself) and a descriptor code, which defines block attributes (last block in the file, last block in the record or restart marker). The FTP protocols do not impose any restrictions on either data type or structure used with block mode transfers. The actual FTP implementations do, however, impose restrictions of various kinds. In CS for z/OS IP, for example, block mode transfer is only supported with a data type of EBCDIC. You may use block mode when you transfer files between z/OS hosts. A file transferred between two MVS systems in block mode preserves its record structure unchanged, including files with variable length records

Compress Data is transmitted in a compressed format. The compression algorithm is rather simple. It includes the ability to send replicated bytes in a two-byte sequence (maximum 128 replications), and to send a filler byte in a one-byte sequence (maximum 64 filler bytes). A filler byte is defined as *space* for ASCII or EBCDIC data types and as binary zero for a data type of image. In CS for z/OS IP compressed mode requires a data type of EBCDIC.

Table 3-2 provides an overview of the supported combinations of data type, structure, and mode. Table 3-2 also provides cross-references between mode, type, and structure. The various options are also discussed in greater detail in the following sections.

Table 3-2 Cross-references between mode, type, and structure

	Data type			**Data structure**	
Transfer modes	**ASCII**	**EBCDIC**	**Image**	**File**	**Record**
STREAM	Yes	Yes	Yes	Yes	Yes
BLOCK	No	Yes	No	Yes	No
COMPRESSED	No	Yes	No	Yes	No

When you select among the options listed above, you have to consider the purpose of your file transfer:

- Are you going to transfer a file to a host, where the file will be processed by programs on that host? In that case, you must select options that will result in a file that can be used by the target host. If the data is text, the originating host uses EBCDIC, and the target host uses ASCII, you must use an ASCII data type and a stream transfer mode.
- Are you going to transfer a file to another host for intermediate storage only and later retrieve it again on the original host? In this case it is very important that you can invert the process, so the file you will end up with back on your original host is exactly like the file you started with. If it is text data, you may not need to translate between EBCDIC and ASCII, and you can use the BINARY data type instead.

FTP LOCSTat and STAtus subcommands

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

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

Example 3-3 Use of STAT and LOCSTAT commands.

```
ftp> quote stat (autor                1
211-Automatic recall of migrated data sets.
211 *** end of status ***
ftp>

EZA1460I Command:
EZA1736I   STAT (INACTIVETIME         2
EZA1701I >>> XSTA (INACTIVETIME
211-Inactivity timer is set to 300
211 *** end of status ***

EZA1460I Command:
EZA1736I   LOCSTAT DCONNTIME          3
EZA2811I DCONNTIME is 120
```

1 The STAT command issued to a z/OS server from a remote non-z/OS client that does not know what the STAT command is must send the command to the server using the QUOTE command.

2 A local z/OS client sends the STAT command to a z/OS server

3 The LOCSTAT command being sent to the local z/OS client

Transferring MVS data sets to stream-oriented file systems

If you want to use a stream-oriented file system as intermediate storage for a record-oriented MVS file then you may have a problem, depending on the record format of the MVS data set you want to store and the data type you use. For ASCII data types, record boundaries do not impose problems. The record boundaries are preserved by means of CRLF (Carriage Return, Line Feed - X'0D0A') for DOS-based systems[1] or just LF (Line Feed - X'0A') for UNIX-based

[1] Including all the descendants of DOS.

systems. If such a data set is transferred from MVS to, for example, UNIX and back to MVS again, the CRLF or LF is used to rebuild the record boundaries and the data set will be identical to the one you originally had on MVS. This is true for both fixed length and variable length record data sets.

For BINARY or IMAGE transfer from MVS to a stream-oriented file system, the situation is slightly more complicated. When the records of an MVS data set are stored in a stream-oriented file system, the records are stored one after the other as one long stream of bytes, without any notion of the record boundaries from the MVS system.

If the original data set was a fixed length record data set, you can reconstruct this data set if you transfer the file back to MVS using the same logical record length as the original data set. The long stream of bytes is chopped into records of exactly the length you specify, thereby reconstructing the same record boundaries as you had in the original data set.

If the original data set was a variable length record data set or a data set with undefined record format, you cannot use the above technique since no knowledge length of each record in the original data set has been retained. There are two ways in which you can move a variable record length file out of MVS and back into MVS again, preserving the record boundaries:

- Use the RDW option of the z/OS FTP client or z/OS FTP server.
- Use the record structure option.

Note: We strongly recommend that you use the record structure option. This is one of the standard file structures defined in RFC959. Both the FTP server and FTP client support it.

Using the RDW option

If you use the FTP client or the FTP server to transfer a variable length record data set from MVS to a stream-oriented file system and you use the RDW option, the file stored on the stream-oriented file system will include the record descriptor words (RDWs) of each record in the original data set.

If the purpose of including the RDWs was to let an application program on the remote host work with the information in the file including the RDWs, you have accomplished what you wanted, but there may be situations in which you might want to get such a file back into MVS again. Unfortunately, the code in CS for z/OS IP that stores the data set on MVS DASD does not use the preserved RDWs to reconstruct record boundaries. Instead, the DCB information given in either the SITE or the LOCSITE command is used. You can implement a solution to this problem in the following way:

1. Use the z/OS FTP client or the z/OS FTP server to transfer a RECFM=V or VB data set to Linux, for example, using the BINARY, STREAM, and RDW option. This will give you the file on Linux with imbedded RDWs.
2. Transfer the file back to MVS using the BINARY, STREAM, and MVS SITE parameters of RECFM=U and BLKSIZE=some high value.
3. Create a program that, based on the imbedded RDWs, reconstructs the original record structure.
4. Be careful using the RDW option with ASCII transfers. Transferring the file out of MVS will work without problems, but if you later want to transfer the file back into MVS, the ASCII-to-EBCDIC translation will also translate the RDWs, which may have unexpected results.

Using the FTP record structure option

When you connect an FTP client to the z/OS FTP server, you can use the record structure option to transfer a variable length record data set from MVS to a stream-oriented file system without the need to deal with RDWs.

Retrieving a recfm=vb data set from the z/OS FTP server to a non-z/OS client

To do this:

1. Connect your FTP client to the z/OS FTP server and set transfer mode to binary.
2. Issue a **dir** command and make a note of the record format (which should be VB), record length, and block size. You need this information later if you want to return this data set back to the z/OS FTP server.
3. Issue a **quote stru r** command. This command is not interpreted by the FTP client, but sent directly to the z/OS FTP server. The effect of this command is that the z/OS FTP server sends data to the FTP client with imbedded end-of-record sequences.
4. Issue a **get** command to copy your variable record length file from z/OS to the client. Because the **structure** command was sent in quotation marks, the client does not know about it and will receive and store the file as a binary stream of bytes, including the imbedded EOR sequences. When you want to copy the file back into MVS again, connect your FTP client to the z/OS FTP server, set transfer mode to binary, and send the **quote** command with a **structure** operand telling the z/OS FTP server to expect records with imbedded EORs.

Sending a recfm=vb data set from a non-z/OS client to a z/OS FTP server

To do this:

1. Connect your FTP client to the z/OS FTP server and set transfer mode to binary.
2. Issue a **quote stru r** command. This command is not interpreted by the FTP client, but is sent directly to the z/OS FTP server. The effect of this command is that the z/OS FTP server receives data from the FTP client and recognizes the embedded end-of-record sequences.
3. Depending on your default SITE parameters, you may need to send a SITE command with record format and record length information to the z/OS FTP server before you issue your **put** for the file.
4. Issue a **put** command. Since the FTP client does not know about the record structure, it transfers the file as a stream of bytes. The file still has the imbedded EOR sequences that are interpreted by the z/OS FTP server to rebuild the original record boundaries.

While the technique above can be used to transfer the VB data set in binary mode, it is still difficult to use the contents of a file at the remote system, because the file received contains imbedded EOR sequences. Any manipulation of the file on the remote server must be careful to preserve the format of the file.

The FTP client included in the z/OS Communications Server can support the record structure option. Therefore you can easily transfer any VB files via structure mode between MVS systems.

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

To do this:

1. Connect the z/OS FTP client to the z/OS FTP server and set transfer mode to binary.
2. Issue a **stru r** command. Since both FTP server and client can recognize the **stru r** command, you do not need to add the **quote** command in front for a z/OS-to-z/OS transfer. Both FTP server and client will recognize the file structure as record.

3. Issue a **put** or **get** command.

FTP UTF-8 data transfer and storage

UNICODE provides a unique number for every character, no matter what platform, program or language you are using. The same code page (UNICODE) is supported regardless of which operating system it runs on (z/OS, Windows®, Linux, AIX®, and so on). When storing UNICODE files you have the choice of retaining, discarding, or creating a byte order mask. This facilitates upload of UNICODE files from workstations to mainframe print solutions that support UNICODE.

Within UNICODE, there are multiple encoding methods. This function addresses only UTF-8 encoding. This implementation of UNICODE is a step forward in moving z/OS to fuller Unicode enablement.

Considerations for using UTF-8 support

For our usage we needed to observe the following points:

- This implementation will be used to transfer UTF8-encoded data from other platforms to z/OS and to maintain its characteristics solely for the purpose of printing UTF-8 encoded files on z/OS system printers.
- Errors in the UTF-8 stream cannot be detected by FTP when you specify MBDATACONN=(UTF-8,UTF-8) to transfer UNICODE files.
- This implementation enables the use of z/OS systems as data repositories for UNICODE data, as well as the use of a high speed printing system as a printing solution for UNICODE data.
- The FTP client must provide support for transferring files using UNICODE when you use a **get** to store the files in your client.
- z/OS FTP allows users to specify whether files stored as UTF-8 are stored with a byte order mask.
- The purpose of a byte order mask is to indicate whether a UNICODE data stream is encoded with “Little Endian” or “Big Endian”. It refers to the byte order in which multiple numbers are stored.
  - “Little Endian” means that the low-order byte of the number is stored in memory at the lowest address, and the high order byte at the highest address.
  - “Big Endian” means that the high-order byte of the number is stored in memory at the lowest address, and the low-order byte at the highest address.
- For UTF-8, the value of the byte order mask is EF BB EF. This sequence can appear anywhere in the file, but it is considered a byte order mask only when it appears in the first character position of the file.

Configuration of UTF-8 support

We used a Windows system to generate the UNICODE file and transfer it to the z/OS FTP Server, which supported the UTF-8 encoding data transfer. One might think that we could transfer UTF-8 simply by setting TYPE to IMAGE (binary) before sending the files. However, a binary transfer does not translate newline markers to EOL when sending, and the EOLs are not translated to newline markers when receiving.

The use of FTP UTF-8 is invoked by using the following statements and subcommands:

```
ENCODING=MBCS
TYPE=ASCII
MBDATACONN=(UTF-8,UTF-8)
UNICODEFILESYSTEMBOM
```

```
MBREQUIRELASTEOL
```

These statements and subcommands may be coded in the z/OS FTP server FTP.DATA file, the z/OS FTP client FTP.DATA file, or issued via the SITE command for the server or the LOCSITE subcommand for the client. Type, ENCODING and MBDATACONN are existing FTP configuration options and subcommands.

UNICODEFILESYSTEMBOM is used to specify whether to add a Byte Order Mark (BOM) to a file stored in the local file system when the file system code page is UNICODE. The options are:

- ASIS: if a Byte Order Mark is present in a UNICODE file that is received from the network, store the file with a Byte Order Mark. If a Byte Order Mark is not present, store the file without a Byte Order Mark. This option is the default.
- ALWAYS: always include a Byte Order Mark when storing the file. If the file is received without a Byte Order Mark, insert a Byte Order Mark into the file.
- NEVER: never include a Byte Order Mark when storing a UNICODE file. If the file is received with a Byte Order Mark, discard it before storing the file.

When appending to a nonexistent file, the FTP server respects the UNICODEFILESYSTEMBOM setting. However, when appending to an existing file, the FTP server always strips a leading Byte Order Mark from the incoming file. This prevents a superfluous BOM from being inserted in the midst of a server file.

MBREQUIRELASTEOL is used to specify whether FTP will require the last record of incoming multibyte files to end with the FTP standard EOL sequence. This setting applies when the server is receiving a multibyte file from the client, as well as when the client is receiving a multibyte file from the server. If you set MBREQUIRELASTEOL to FALSE, and you have coded CHKCONFIDENCE TRUE in FTP.DATA, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record will be high. If you set MBREQUIRELASTEOL to TRUE, and you have coded CHKCONFIDENCE TRUE in FTP.DATA, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record is low.

MBDATACONN is used to define the conversions between file system code page and the network transfer code page during data transfer. It affects the conversion of double-byte character set (DBCS) and multibyte character set (MBCS) data and is used when the ENCODING MBCS statement is coded. The SITE and LOCSITE subcommands are also available to set this keyword. The options (UTF-8,UTF-8) are used to support UNICODE encoding. Example 3-4 shows the use of these parameters.

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

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

To create UTF-8 file on our workstations we used the Windows notepad to edit a file and save it with the appropriate encoding option, as shown in Figure 3-3.

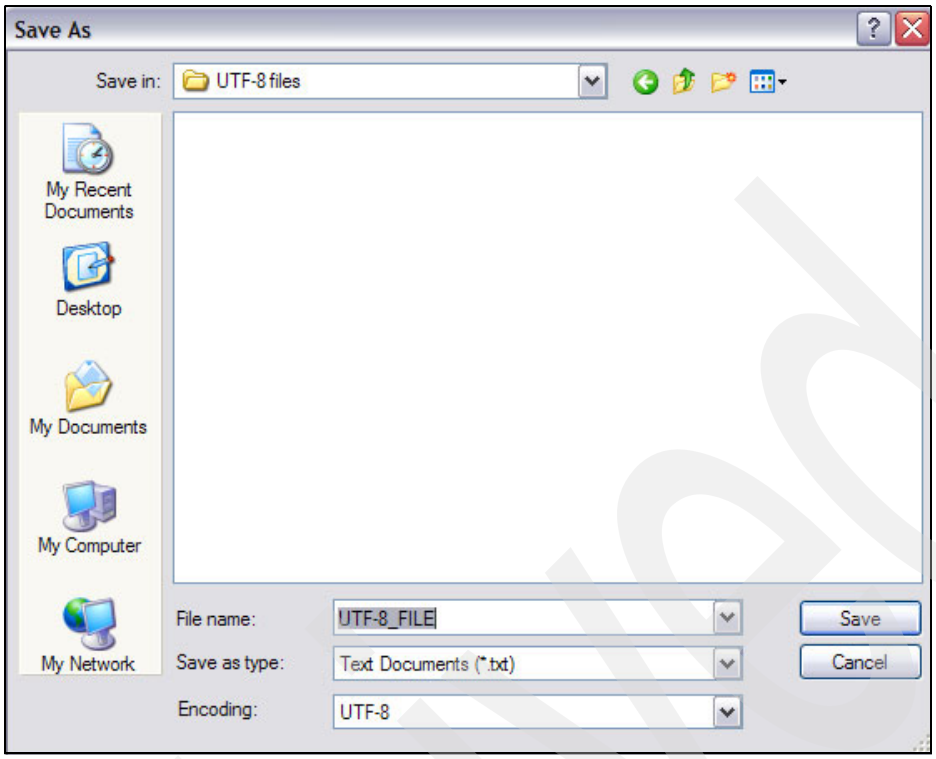


Figure 3-3 UTF-8 file encoding

Example 3-5 illustrates the complete FTP dialog for the file.

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

```
C:\>ftp 10.1.1.20
Connected to 10.1.1.20
220-FTPDD1 IBM FTP CS at WTSC31B.itso.ibm.com, 21:40:14 on 2007-09-23.
220 Connection will close if idle for more than 5 minutes.
User (10.1.1.20:(none)): cs07
331 Send password please.
Password:
230 CS07 is logged on.  Working directory is "CS07.".
ftp> ascii                                                  1
200 Representation type is Ascii NonPrint
ftp> quote site encoding=mbcs                               2
200 SITE command was accepted
ftp> quote site mbdataconn=(utf-8,utf-8)                    3
200 SITE command was accepted
ftp> quote site unicodefilesystembom=asis                   4
200 SITE command was accepted
ftp> put c:\UTF-8_FILE.txt 'TCPIP.ITSO.FTPUTF8'             5
200 Port request OK.
125 Storing data set TCPIP.ITSO.FTPUTF8
250 Transfer completed successfully.
ftp: 121 bytes sent in 0.00Seconds 121000.00Kbytes/sec.
ftp>
ftp> quote stat                                             6
211-using Mode Stream, Structure File, type ASCII, byte-size 8
211-ENcoding is set to MBCS
211-MBdataconn codeset names: utf-8,utf-8
211-Server site variable MBREQUIRELASTEOL is set to TRUE
211-Server site variable UNICODEFILESYSTEMBOM is set to ASIS
```

```
ftp>
```

1 setting ASCII mode

2 setting encoding to mbcs

3 setting UTF-8 encoding

4 setting the UnicodeFileSystem type

5 transferring the UTF-8 file

6 displaying the settings to verify proper values

Dependencies for UTF-8

When a multibyte character set (MBCS) file is created without an EOL <crlf>, setting MBREQUIRELASTEOL on will cause the FTP file transfer to fail, as shown in Example 3-6.

Example 3-6 Attempt to transfer a file without EOL in the last record.

```
ftp> put c:\CSFTPUTF 'TCPIP.ITSO.FTPUTF8'
200 Port request OK.
125 Storing data set TCPIP.ITSO.FTPUTF8
451 Transfer aborted due to file error. File is catalogued.                    1
451-File Transfer might be incomplete. Last record received without EOL sequence
ftp: 125 bytes sent in 0.00Seconds 125000.00Kbytes/sec.
```

1 It would appear the file transfer failed, (message 451 implication) but actually the file has been stored in the dataset. The problem is that Windows sent the file without an EOL sequence in the final record.To correct this without specifying MBREQUIRELASTEOL, edit the file with an ASCII-based editor and scroll to the end of the data. Press <Enter> and save the file. Transmitting it again should work. If a file contains an EOL <crlf> before the actual end of the data, FTP transfer will only transmit data up to that EOL and MBREQUIRELASTEOL will return a successful return code.

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

FTP Kerberos single sign-on support

The FTP server supports single sign-on for Kerberos connections. A Kerberos-based network authentication service enables applications that use Kerberos to use a Kerberos ticket for authentication instead of a user ID and password. This enables you to sign on once to a Kerberos-based security server and not be prompted for a password when accessing the FTP server.

The FTP server statement, SECURE_PASSWORD_KERBEROS with a setting of OPTIONAL permits single sign-on for Kerberos connections.

The FTP client can set the source IP address

The FTP client supports a command line parameter, *-s srcip*, to specify the source IP address that the FTP client should use for client initiated connections to the server. This specification

overrides the source IP address that the stack would ordinarily assign to the outbound packets that belong to this client-initiated connection.

TLS and SSL RFC level support

You can configure z/OS FTP to conform to either the draft level or to the RFC 4217 level of securing FTP with TLS. This is specified by coding the TLSRFCLEVEL statement with either a setting of DRAFT or RFC4217.

FTP reporting of incorrect message catalog levels

FTP verifies that the dates within its z/OS UNIX message catalogs, *ftpdmsg.cat* and *ftpdrply.cat*, match the timestamps of the message catalogs used to produce its default message texts. This reduces the chance that FTP will present the wrong message or reply to the user because of a lack of synchronization between message catalogs and FTP load modules.

FTP client support for sequence numbers in the INPUT file

The FTP client, by default, treats sequence numbers present in the INPUT command stream as part of the command. However, these sequence numbers can be removed by the client before processing, by specifying SEQNUMSUPPORT TRUE. This reduces the likelihood that sequence numbers in that input will be interpreted as part of the command to the FTP client. There are specific requirements for this statement when the INPUT consists of multiple, concatenated datasets.

3.2.3 Configuration of basic FTP without security

In this section we set up an FTP server and FTP client with settings we would expect to encounter in a typical installation. We set up the FTP server in one LPAR, the FTP client in a different LPAR, and then use the FTP client to connect to the FTP server.

In order to use FTP we need to perform the following tasks.

- Prepare the client environment
- Prepare the server environment

Prepare the client environment

In the LPAR used for the FTP client, you need to create an FTP.DATA for the client. We copied the sample FTP.DATA for the client from SEZAINST(FTCDATA) to our TCPIPB.TCPPARMS. Next, determine if you need to further customize FTCDATA for your installation. *z/OS Communications Server: IP Configuration Reference*, SC31-8776, contains a description of all the FTP.DATA parameters. We used the supplied FTCDATA without any additional changes.

The FTP client uses the search order documented in *z/OS Communications Server: IP Configuration Reference*, SC31-8776, to locate its FTP.DATA data set. We recommend that you use a SYSFTPD DD card in your TSO logon procedure, or use the -f parameter to point to the desired configuration file. This ensures that you are using the correct FTP.DATA data set. Your FTP client is now ready for use.

Prepare the server environment

Perform the remaining tasks on the LPAR used for your FTP server:

- Define SAF definitions for the FTP server
- Create the FTP.DATA for the server
- Create the catalogued procedure for the FTP server

- Create the STDENV file for the FTPD server
- Add FTP to the AUTOLOG and PORT statements
- Update the FTP port entry in /etc/services
- Configure syslogd

Define SAF definitions for the FTP server

At a minimum, the following definitions are required to start and use FTP:

- The FTP catalogued procedure must be defined to the security program and added to the started class facility or started procedures table. The user ID associated with the task must have a UID of 0. If the FACILITY class is active, then the FTP user ID requires READ access to the BPX.DAEMON or BPX.POE profiles, if defined. If the FTP address space is to be nonswappable, then the user ID also requires READ access to FACILITY class resource BPX.STOR.SWAP. We used the same ID used by the TCP/IP stack to start FTPD.
- Any user that will log into FTP must have an OMVS segment defined in the security profile for that user ID.

Important: There may be additional security requirements in your environment if EZB.STACKACCESS, EZB.PORTACCESS, or EZB.NETACCESS controls are in use. *z/OS Communications Server: IP Configuration Guide*, SC31-8775, lists the optional security requirements. Access controls are discussed in *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535.

Create the FTP.DATA for the server

First, copy the sample FTP.DATA for the server from SEZAINST(FTPSDATA) to your TCPPARMS. Next, determine if you need to further customize FTPSDATA for your installation. The *z/OS Communications Server: IP Configuration Reference*, SC31-8776, contains a description of all the FTP.DATA parameters. We used the sample FTPSDATA member to create our own member called FTPDBxx (where xx is the &sysclone system symbolic), and simply uncommented the BANNER and ADMINEMAILADDR statements. The changes we made to the file for our FTP server installation are shown in Example 3-7. Neither parameter is crucial to the execution of the server, we used them to simply show how to make changes to the file.

Example 3-7 Changes to FTPDBxx which was copied from FTPSDATA

```
BANNER /etc/ftpbanner
ADMINEMAILADDRESS user@host.my.net
```

The contents of our /etc/banner file are shown in Example 3-8. Note the /etc/ftpbanner file needs to be readable by the user ID of the FTP server proc. The user ID starting FTP is a root user so file permissions 700 (rwx------) provide sufficient access. In our case this user ID is TCPIP.

Example 3-8 /etc/banner

```
****************************************
 Welcome to the ITSO FTP server

 This system may be used for
management approved purposes only.

 All transfers are logged.
```

```
 Please report problems to %E
****************************************
```

Create the catalogued procedure for the FTP server

Copy the sample FTP procedure from SEZAINST(FTPD) to your PROCLIB and customize the sample. The parameters on the EXEC and the SYSFTPD DD statements need to be updated. Ensure that SYSFTPD refers to the correct FTP.DATA for the server and that the RESOLVER_CONFIG environment variable points to the resolver configuration data set you want to use. We chose to use the _CEE_ENVFILE environment variable to point to a DD statement which defines a dataset containing other environment variables for the server. Our procedure is shown in Example 3-9.

Example 3-9 Catalogued procedure for the FTP server

```
//FTPDB  PROC PARMS='',STDENV=FTPENV&SYSCLONE.,FTPDATA=FTPDB&SYSCLONE. 1
//FTPDB  EXEC PGM=FTPD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/&PARMS')
//STDENV  DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV.)   2
//SYSFTPD DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&FTPDATA.)            3
//CEEDUMP  DD SYSOUT=*
```

The following items explain Example 3-9:

- 1 The STDENV symbolic parameter specifies the member name containing environment variables. We made use of the &SYSCLONE system symbolic to determine the name of the member. The FTPDATA symbolic parameter specifies the member name of the FTP.DATA configuration file. We made use of the &SYSCLONE system symbolic to determine the name of the member. Using symbolic parameters on the PROC statement enables more flexibility in starting the procedure.
- 2 The standard environment variable file must have RECFM =V or VB. The specified member contains environment variable settings.
- 3 The FTPDATA member contains FTP client configuration statements to override internal defaults.

Create the STDENV file for the FTPD server

Example 3-10 shows the STDENV environment variable file we used for our FTPD server.

Example 3-10 FTP server standard environment variable file

```
BROWSE    TCPIP.SC31.STDENV(FTPENV31) - 01.00        Line 00000000 Col 001 076
********************************* Top of Data **********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'  1
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB              2
_BPX_JOBNAME=FTPDB                            3
TZ=EST                                        4
******************************** Bottom of Data ********************************
```

The following items explain Example 3-10:

1 The resolver needs access to the TCPDATA file for functions such as DNS name resolution.

2 We set stack affinity to our local stack. This is necessary only if multiple stacks exist on the LPAR.

3 The initial job name of the parent process is always FTPDB1 (the proc name suffixed with the number 1) regardless of the jobname setting. We set _BPX_JOBNAME so that all subsequent FTP forked instances that are created by a client connection have the *same* job name: FTPDB. Setting the job name in this way enables syslogd *isolation to function* and sends all FTP log messages to a single file, as discussed in Chapter 1, “Syslogd” on page 1. If you do not set the _BPX_JOBNAME environment variable, then at each subsequent fork, the job name becomes FTPDB*x,* where *x* is a number between two and nine.

4 We set the TZ environment variable so that messages issued to the console or to SYSLOGD are in the correct time zone.

4 We point the server to the FTP.DATA configuration file that we prepared.

> **Note:** You can use the *_CEE_ENVFILE* environment variable in the PARM field of the JCL to point to a file that contains other environment variables. The file can be a UNIX file, a zFS, or a z/OS MVS dataset. When it is an MVS data set, the data set must be allocated with RECFM=V.
>
> RECFM=F must *not* be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in UNIX System Services. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Add FTP to the AUTOLOG and PORT statements

AUTOLOG allows you to automatically start the FTP server when the TCP/IP stack has initialized, and to automatically restart it anytime the FTP server fails. To enable this support you need to add an AUTOLOG statement for the FTP server. AUTOLOG needs to know the name of the procedure that starts FTPD and needs to know the job name of the running FTP server process in order to monitor it. The FTP server forks on startup, so the actual job name of the running FTP server may be different from the original job name assigned when the FTP server was started. In our environment, we start the FTP server with a job name of FTPDB, and expect the first forked task to be named FTPDB1. FTPDB1 is the name of the FTP server listener. All subsequent FTP server address spaces that represent logged-in users will be named FTPDB because we have set the _BPX_JOBNAME variable to force all FTP jobs to be named FTPDB.

The PORT statement in PROFILE.TCPIP should reserve the FTP control and FTP data ports to the FTP job name. The default ports for FTP are TCP port 21 for the control connection and TCP port 20 for the data connection. The port must be reserved to the job name of the running FTP server. Example 3-11 shows our AUTOLOG and PORT statements in PROFILE.TCPIP.

Example 3-11 Statements in PROFILE.TCPIP for the FTP server

```
AUTOLOG
  FTPD JOBNAME FTPDB1

PORT
  20 TCP  FTPDB NOAUTOLOG
  21 TCP  FTPDB1
```

Update the FTP port entry in /etc/services

/etc/services maps service names to port names. Unless overridden by a command line parameter, when the FTP server starts it searches /etc/services for the service name *ftp* and

attempts to bind the FTP control socket to the port associated with the FTP service. Our /etc/services is shown in Example 3-12.

Example 3-12 FTP entry in /etc/services

```
ftp  21/tcp
```

Configure syslogd

We highly recommended that you configure syslogd before starting the FTP server. If you use our recommended syslogd configuration described in Chapter 1, “Syslogd” on page 1, then your syslogd configuration file already contains one of the lines shown in Example 3-13. This will capture all messages from the FTP server task that has a job name that starts with FTPDB to a separate log file named *ftpd.log*. Refer to Chapter 1, “Syslogd” on page 1, for more details on syslogd.

Example 3-13 Line in /etc/syslog.conf for the FTP server

```
*.FTPDB*.*.* /var/log/%Y/%m/%d/ftpd.log              or
*.FTPDB*.*.* /tmp/syslog/%Y/%m/%d/ftpd.log      or
*.FTPDB*.*.* /tmp/syslog/ftpd.log
```

3.2.4 Activation and verification for basic FTP without security

When all the preparation tasks have been completed, you are ready to start the FTP server using your started procedure.

Start the FTP server

```
S FTPDB
```

There are a number of ways to verify that the FTP server has started and is working correctly. First, we look for message EZY2702I on the system console.

Check startup message for FTP server

Example 3-14 shows the EZY2702I message we received shortly after FTP was started.

Example 3-14 EZY2702I message received on successful startup of FTP

```
EZY2702I Server-FTP: Initialization completed at 21:29:05 on 09/23/07.
```

Verify that FTP server is functional before client login

To verify that the FTP server is functional, we used the z/OS FTP client, testing our FTP client as well. The following tasks show the progression of a client connection and the assignments of jobname and user security context to the forked processes. In sequence, we show:

- a client connection log before user login (before user ID and password submitted)
- an OMVS display of the FTPDB* threads before user login
- an MVS display of the active FTPDB* jobs
- a NETSTAT connection display of the FTPDB* client connections

Check client log for successful connection to server, before client login

Example 3-15 shows the output log of our FTP client from another z/OS LPAR immediately after a connection, but *before* we logged into the server.

Example 3-15 FTP client log before client login

```
FTP: using TCPIPB
```

```
Connecting to:   10.1.1.20 port: 21.
220-FTPDB1 IBM FTP CS at WTSC31B.ITSO.IBM.COM, 21:31:05 on 2007-09-23.  1
220 Connection will close if idle for more than 5 minutes.
NAME (10.1.1.20:CS10):                                                  2
```

1 The 220 server message indicates that the server has accepted the connection

2 The client is prompting the user for the user ID to be passed on to the server

Check OMVS status of FTPDB threads before client login*

Example 3-16 shows the OMVS status of the server listener and the client connection *before* the client supplies user ID and password.

Example 3-16 OMVS display of FTPDB threads before client login

```
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
TCPIP    FTPDB1   0069     131312          1 1FI--- 10.59.38      .039  1
  LATCHWAITPID=         0 CMD=FTPD

TCPIP    FTPDB    004D   84017348     131312 1FI--- 13.57.41      .022  2
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 0 0 27 1 80 128 256 5
```

1 This line shows information for the server's listener:

- The security context (user ID) is TCPIP and always will be because it is the listener's
- The jobname is FTPDB1 and always is because it is the first forked process
- The process ID (PID) is 131312
- The time the address space was created is 10.59.38
- The CPU time accumulation in seconds is .039

2 This line shows information for the client connection:

- The security context (user ID) is TCPIP because the client has *not* logged in, yet
- The jobname is FTPDB because we used _BPX_JOBNAME to force all forked processes to have the same name
- Notice that this entry has its own PID, but also is tied to the Parent PID (PPID) of the server's listener process (131312)
- The time the address space was created is 13.57.41
- The CPU time accumulation in seconds is .022

Check MVS status of FTPDB jobs before client login*

Example 3-17 shows the output from an MVS display of the FTPDB* jobs *before* client login. Both the FTP server listener task and our forked task for the connected client are shown.

Example 3-17 MVS display of active FTPDB jobs before client login*

```
JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
00008    00029    00002     00034    00023    00002/00030       00031
 FTPDB1   STEP1    TCPIP    OWT  AO  A=0069   PER=NO   SMC=000         1
                                     PGN=N/A  DMN=N/A  AFF=NONE
                                     CT=000.048S  ET=03.32.46          2
                                     WUID=STC03783 USERID=TCPIP        3
                                     WKL=SYSTEM   SCL=SYSOTHER P=1
                                     RGP=N/A      SRVR=NO  QSC=NO
                                     ADDR SPACE ASTE=03ED6A40
 FTPDB    STEP1    TCPIP    OWT  AO  A=004D   PER=NO   SMC=000         4
                                     PGN=N/A  DMN=N/A  AFF=NONE
                                     CT=000.014S  ET=007.270S          5
```

```
                                WUID=STC03866 USERID=TCPIP        6
                                WKL=SYSTEM   SCL=SYSOTHER P=1
                                RGP=N/A      SRVR=NO  QSC=NO
                                ADDR SPACE ASTE=03ED6340
```

1 Jobname FTPDB1 running under user ID TCPIP is the FTP server listener. This is always the case, because it is the first forked process.

2 The Elapsed Time field (ET=) indicates *how long* the server's listener connection has been in place.

3 The USERID field indicates the user ID under which the listener connection is running.

4 Jobname FTPDB running under user ID TCPIP is the FTP client connection. This is the forked task that is serving our connected client. It shows the server's user ID until the client logs in with user ID and password, at which time, the security context of the address space will be changed from the server's user ID to the client's user ID

5 The Elapsed Time field (ET=) indicates *how long* the client's connection has been in place.

6 The Userid field indicates the user ID under which the client connection is running. The user ID (security context) will remain the user ID of the FTP server until the client logs in.

Check NETSTAT conn status of FTPDB connections before client login*

Example 3-18 shows the NETSTAT output for connections belonging to FTPDB* *before* client login.

Example 3-18 NETSTAT display of FTPDB connections before client login*

```
D TCPIP,TCPIPB,N,CONN,CLIENT=FTPDB*
FTPDB1   0000144F LISTEN                    1
  LOCAL SOCKET:   ::..21                    2
  FOREIGN SOCKET: ::..0
FTPDB1   0000178C ESTBLSH                   3
  LOCAL SOCKET:   ::FFFF:10.1.1.20..21      4
  FOREIGN SOCKET: ::FFFF:10.1.1.40..1031    5
```

Jobname FTPDB1 shows two connections:

1 The listener, on port 21 (2) in a listen state that represents the listener task

3 The connection between local port 21 (4) and 10.1.1.40 port 1031 (5) that represents the established control connection for our client. Note that the established connection for the client shows job name FTPDB1 (the server listener ID) and not job name FTPDB, as we would expect. This is because the job that originally accepted the connection was FTPDB1 and only *after* the connection was established was a new job forked to handle the FTP session. So, as far as the stack is concerned, the original jobname is what is assigned to the client's control connection. When the data connection is opened for a data transfer, the data connection (on port 20) will be assigned a jobname of FTPDB, as expected.

Verify that FTP server is functional after client login

We show the same displays again, *after* user login:

- a client connection log after user login
- an OMVS display of the FTPDB* threads after user login
- an MVS display of the active FTPDB* jobs
- a NETSTAT connection display of the FTPDB* client connections

Check client log after client login

Example 3-19 shows the output log of our FTP client from the other z/OS LPAR *after* we successfully logged in to our FTP server.

Example 3-19 FTP connect from one z/OS system to another z/OS system: after login

```
FTP: using TCPIPB
Connecting to:   10.1.1.20 port: 21.
220-FTPDB1 IBM FTP CS at WTSC31B.ITSO.IBM.COM, 21:31:05 on 2007-09-23.  1
220 Connection will close if idle for more than 5 minutes.
NAME (10.1.1.20:CS10):
cs10
>>> USER cs10
331 Send password please.
PASSWORD:

>>> PASS
230 CS10 is logged on.  Working directory is "CS10.".                  2
Command:
```

1 The 220 server message indicates that the server on SC30 has accepted the connection.

2 The 230 server message indicates that the server on SC30 has accepted the user ID and password from the client, and has established the default high level qualifier for dataset access to that of the user ID.

Check OMVS status of FTPDB threads after client login*

Example 3-20 shows the OMVS status of the server listener and the client connection *after* the client supplies user ID and password.

Example 3-20 OMVS display of FTPDB threads after client login

```
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
TCPIP    FTPDB1   0069     131312          1 1FI--- 10.59.38      .039     1
  LATCHWAITPID=         0 CMD=FTPD

CS10     FTPDB    004D   84017348     131312 1FI--- 13.57.41      .045     2
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2136705232
```

1 This line shows information for the server's listener: (it has not changed)

- The security context (user ID) is TCPIP and always will be because it is the listener's
- The jobname is FTPDB1 and always is because it is the first forked process
- The process ID (PID) is 131312
- The time the address space was created is 10.59.38
- The CPU time accumulation in seconds is .039

2 This line shows information for the client connection: (some of the info has changed)

- The security context has been changed to CS10, the client has logged in
- The jobname is FTPDB because we used _BPX_JOBNAME to force all forked processes to have the same name
- Notice that this entry has its own PID, but also is tied to the Parent PID (PPID) of the server's listener process (131312)
- The time the address space was created is 13.57.41
- The CPU time accumulation in seconds has changed from .022 to .45, which is attributed to the login process.

Archived

Check MVS status of FTPDB jobs after client login*

Example 3-21 shows output from the D J,FTPDB* command *after* we logged into the FTP server. Both the FTP server listener task and our forked task for the connected client are shown.

Example 3-21 D J,FTPDB after login to FTP server*

```
JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
00008    00029    00002      00034    00023    00002/00030       00031
 FTPDB1   STEP1    TCPIP    OWT  A0  A=0069   PER=NO   SMC=000          1
                                     PGN=N/A  DMN=N/A  AFF=NONE
                                     CT=000.048S  ET=03.33.31            2
                                     WUID=STC03783 USERID=TCPIP          3
                                     WKL=SYSTEM   SCL=SYSOTHER P=1
                                     RGP=N/A      SRVR=NO  QSC=NO
                                     ADDR SPACE ASTE=03ED6A40
 FTPDB    STEP1    CS10     OWT  A0  A=004D   PER=NO   SMC=000          4
                                     PGN=N/A  DMN=N/A  AFF=NONE
                                     CT=000.020S  ET=039.499S            5
                                     WUID=STC03866 USERID=CS10           6
                                     WKL=SYSTEM   SCL=SYSOTHER P=1
                                     RGP=N/A      SRVR=NO  QSC=NO
                                     ADDR SPACE ASTE=03ED6340
```

1 Jobname FTPDB1 running under user ID TCPIP is the FTP server listener.

2 The Elapsed Time field (ET=) indicates *how long* the server's listener connection has been in place.

3 The USERID field indicates the user ID under which the listener connection is running.

4 Jobname FTPDB running under user ID CS10 is the FTP client connection. This is the forked task that is serving our connected client.

5 The Elapsed Time field (ET=) indicates *how long* the client's connection has been in place.

6 The Userid field indicates the user ID under which the client connection is running.

Check NETSTAT conn status of FTPDB connections after client login*

Example 3-22 shows the output from a **`netstat conn`** command after client login.

Example 3-22 Output of netstat conn, after user login

```
D TCPIP,TCPIPB,N,CONN,CLIENT=FTPDB*
FTPDB1   0000144F LISTEN                    1
  LOCAL SOCKET:   ::..21                    2
  FOREIGN SOCKET: ::..0
FTPDB1   0000178C ESTBLSH                   3
  LOCAL SOCKET:   ::FFFF:10.1.1.20..21      4
  FOREIGN SOCKET: ::FFFF:10.1.1.40..1031    5
```

Jobname FTPDB1 shows two connections:

1 The listener, on port 21 (2) in a listen state that represents the listener task

3 The connection between local port 21 (4) and 10.1.1.40 port 1031 (5) that represents the established control connection for our client. Note that the established connection for the client shows job name FTPDB1 (the server listener ID) and not job name FTPDB, as we

Archived

would expect. This is because the job that originally accepted the connection was FTPDB1 and only *after* the connection was established was a new job forked to handle the FTP session. So, as far as the stack is concerned, the original jobname is what is assigned to the client's control connection. When the data connection is opened for a data transfer, the data connection (on port 20) will be assigned a jobname of FTPDB, as expected. See Example 3-23.

Check NETSTAT conn status of FTPDB connections during file transfer*

Example 3-23 shows the output from a **`netstat conn`** command during a client file transfer. When the data connection for the transfer is opened, it will be on port 20, as seen here.

Example 3-23 Output of netstat conn, during a client file transfer

```
D TCPIP,TCPIPB,N,CONN,CLIENT=FTPDB*
. . .
FTPDB    00000197 ESTBLSH                  1
  LOCAL SOCKET:   ::FFFF:10.1.1.20..20     2
  FOREIGN SOCKET: ::FFFF:10.1.1.40..1050   3
FTPDB1   0000144F LISTEN                   4
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDB1   0000178C ESTBLSH                  5
  LOCAL SOCKET:   ::FFFF:10.1.1.20..21
  FOREIGN SOCKET: ::FFFF:10.1.1.40..1031
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

1 The data connection on port 20 is being used for the file transfer and has a jobname of FTPDB which was assigned by the *_BPX_JOBNAME* setting. All forked processes will have this common jobname.

2 The local socket is the IP address of this system that the client used to access this system. Port 20 is used for the data connection.

3 The foreign socket is the IP address of the client which is using ephemeral port 1050 for its data connection.

4 The same original server listener connection on port 21, listening on INADDR_ANY

5 The same original client control connection on port 21 as before

Check server parameter settings with the STAT command

The STAT command is issued by the client to request a listing of the server options. The server responds with reply code 211 messages. We show an abbreviated list of the server options in Example 3-24.

Example 3-24 STAT command shows server option settings

```
>>> STAT
211-Server FTP talking to host ::ffff:10.1.1.40, port 1031          1
211-User: CS10 Working directory: CS10.                             2
211-The control connection has transferred 1108 bytes               3
211-There is no current data connection.
. . .
211-RDWs from variable format data sets are discarded.              4
. . .
211-ENcoding is set to SBCS                                         5
211-Outbound SBCS ASCII data uses CRLF line terminator
211-Outbound MBCS ASCII data uses CRLF line terminator
```

```
211-Server site variable MBREQUIRELASTEOL is set to TRUE
211-Server site variable UNICODEFILESYSTEMBOM is set to ASIS
. . .
211-TLS security is supported at the DRAFT level                    6
. . .
211-Server site variable LISTSUBDIR is set to TRUE                  7
211 *** end of status ***
```

1 The server confirms the client's IP address and port being used

2 The user ID and current working directory are indicated

3 Information the control connection and the data connection is listed

4 Information on how RDWs are treated for RECFM=VB records

5 Single byte and Multi-byte encoding settings are available for display

6 TLS security level support is shown (TLSRFCLEVEL is DRAFT or RFC4217)

7 LISTSUBDIR controls how file names within subdirectories are displayed

Check client parameter settings with the LOCSTAT command

The LOCSTAT command is issued by the client to request a listing of the local client options. The client program responds with a list of option settings. We show an abbreviated list of the client options in Example 3-25.

Example 3-25 LOCSTAT command shows client option settings

```
>>> LOCSTAT
 Trace: FALSE, Send Port: TRUE
 Send Site with Put command: TRUE
 Connected to:10.1.1.20, Port: FTP control (21), logged in          1
 Local Port: 1031
 Data type:a, Transfer mode:s, Structure:f
. . .
 Data connections for the client are not firewall friendly.
 local site variable EPSV4 is set to FALSE
 local site variable SECUREIMPLICITZOS is set to TRUE
 local site variable TLSRFCLEVEL is set to DRAFT                   2
 local site variable LISTSUBdir is set to TRUE
 local site variable PROGRESS is set to 10
 local site variable SEQNUMSUPPORT is set to FALSE                 3
 Authentication mechanism: None
. . .
 Using FTP configuration defaults.                                 4
```

1 The current connection information includes the IP address and port number of the server (10.1.1.20..21), as well as the local ephemeral port of the client (1031).

2 The TLSRFCLEVEL setting at the client can be either DRAFT or RFC4217

3 SEQNUMSUPPORT indicates whether the client supports the presence of sequence numbers in the INPUT command stream. If supported (TRUE), the client program removes them before processing the data. If not supported (FALSE), the sequence numbers are assumed to be part of the input commands.

Archived

4 This line lets you know from what source the FTP client initially gets its option settings. If an FTP.DATA file was specified to the client, its dataset name is listed, otherwise internal hardcoded defaults are used for all the client options. The LOCSITE command can override the initial settings during the session.

3.3 Multiple FTP servers in a sysplex

This section provides an overview of executing multiple FTP servers in a sysplex (one server per LPAR, one stack per LPAR) and using sysplex distribution to load balance between them. Based upon installation policies, the sysplex distributor directs connections to the best available FTP server.

The following topics are discussed:

- Description of multiple FTP servers in a sysplex
- Configuration for multiple FTP servers in the sysplex
- Activation and verification of FTP servers within sysplex

3.3.1 Description of multiple FTP servers in a sysplex

For this situation we use two system images, each with only one TCP/IP stack and only one FTP server associated with it. The TCP/IP stack started task name is the same on both systems: TCPIPB. The FTPD server started task name is the same on both systems: FTPDB. We use system symbolics in the started task JCL to provide uniqueness where necessary.

On system SC30 we have:

- TCPIPB
- FTPDB

On system SC31 we have:

- TCPIPB
- FTPDB

We designed the two stacks to back each other up. We also designed the two FTP servers to back each other up. Even though it is not a requirement for a backup stack to distribute connections identically to the method used in the primary stack, we designed our two stacks to do so. If the primary stack fails, or otherwise relinquishes its distributor responsibilities, our backup stack will continue to distribute connections to the FTPD servers in the same fashion as the primary.

Dependencies of multiple servers within a sysplex

In order to use FTP in a sysplex distributor environment, you need to configure FTP on each LPAR in the sysplex using the directions discussed in 3.2, “Basic FTP without security” on page 84.

Stack dependencies for multiple FTPD servers in sysplex

Because sysplex distribution (SD) is being used in this scenario, all the functionality that a TCP/IP stack needs to support SD is required. Refer to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534, for a discussion on setting up a TCP/IP stack to support sysplex distribution. These requirements include:

- The system hardware and software required to support the Coupling Facility and XCF.
- XCFINIT=YES in VTAM, DYNAMICXCF in TCP/IP.

- An IP subnet and host IP address assigned to the XCF interfaces.
- If HiperSockets are implemented, HiperSockets used by XCF must be consistent in their definitions
- Most of the parameters within GLOBALCONFIG, IPCONFIG, TCPCONFIG, and UDPCONFIG should be set the same on all participating stacks.

The multiple stacks must have Distributed Dynamic VIPA definitions added to support distribution to the multiple FTPD servers. The VIPADYNAMIC block must be coded in the backup stack in such a way that it will distribute connections in a similar manner as the primary, but not necessarily in an identical manner. Our scenario calls for an identical process.

The introduction of sysplex distribution adds the requirement for a new IP subnet for Distributed DVIPA as though Dynamic VIPA has not already been implemented.

FTP server dependencies for multiple servers in sysplex

Because the two FTP servers back each other up, they should be identically configured so they can treat all client connections the same. If they differ in any way clients could experience differences between their FTP sessions.

Advantages of multiple servers within a sysplex

Sysplex distribution provides multiple redundant resources that provide high availability. In our scenario all of the following are redundant resources participating in the high availability that sysplex distribution provides:

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

Sysplex distribution working with Workload Manager provides load balancing between the stacks and between the FTPD servers.

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

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Considerations for running multiple servers within a sysplex

Implementing multiple redundant TCP/IP stacks and multiple redundant FTP servers adds to the clerical effort of systems personnel in maintaining equivalent configurations across all participating systems. The effort in keeping the configurations synchronized is sometimes underestimated or overlooked.

Planning and design is more complex, involving multiple departments. The necessity for cooperation between departments is sometimes underestimated or even unplanned. Mainframe systems and networking personnel must be aware of what the physical network requirements are.

The demand on IP subnets and IP addressing is increased by introducing sysplex distribution, Dynamic VIPAs, and Dynamic XCF.

Operations must be made aware of changes that sysplex distribution, multiple stacks, and multiple server applications introduce to the environment.

Automations and scheduling changes may be required. These issues are sometimes overlooked.

3.3.2 Configuration for multiple FTP servers in the sysplex

Multiple FTP servers can accept connections that are distributed by sysplex distribution. In our environment we set up a sysplex using two stacks, both with the job name of TCPIPB that are in two LPARS, SC30 and SC31. The same sysplex environment is discussed in more detail in 2.3, “Multiple TN3270E servers in a multiple image environment” on page 52.

In order to load balance FTP in a sysplex we need to first perform all the implementation tasks for both client and server in 3.2.3, “Configuration of basic FTP without security” on page 97, or, if security is desired, perform all the secure FTP tasks in *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535. Then we need to perform these additional tasks:

- Customize a second TCP/IP stack started task procedure.
- Customize both TCP/IP stack profile data sets for the sysplex.
- Customize a second FTP server started task procedure.
- Customize a second FTP server configuration data set.
- Customize an OMPROUTE started task to support the second stack
- Customize an OMPROUTE configuration for the second OMPROUTE

Customize a second TCP/IP stack started task procedure

This second started task is for running our sysplex distribution backup stack. It can be modeled after our first (primary) started task. We used the same procedure name and same procedure library for both stacks, and used system symbolics to provide unique names for the stacks’ configuration profile data sets. Our first TCP/IP task is TCPIPB in LPAR CS30 and our second TCP/IP task is also named TCPIPB and is running in LPAR SC31. The TCP/IP started procedures are identical and use system symbolics to identify different TCPIP.DATA and PROFILE.TCPIP data sets.

Customize both TCP/IP stack profile data sets for the sysplex

The stack in LPAR SC30 is our sysplex distributor stack, and the stack in LPAR SC31 is the backup stack. Both stacks are FTP targets. For a complete discussion and examples of setting up a stack, refer to:

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

We need to add statements that enable sysplex functions, create dynamic XCF interfaces, and create dynamic VIPAs. The statements necessary to enable sysplex distribution on TCPIPB on LPAR SC30 are shown in Example 3-26.

Example 3-26 Sysplex enablement for TCPIPB on SC30

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60                    1
;
IPCONFIG
   SYSPLEXROUTING                                                      2
   DYNAMICXCF 10.1.7.11 255.255.255.0 8                                3
;
```

```
VIPADYNAMIC

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.25   ;FTP            4
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM               5
                         10.1.8.25    PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21  ;FTP General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.1.8.21   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM           -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22  ;FTP Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.1.8.22   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using SERVERWLM         -
  ;-------------------------------------------------------------------
   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23  ;FTP Payroll
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21
  ;-------------------------------------------------------------------
  ;      Distribute to 10.1.7.11 via normal XCF   (no viparoute)  -
  ;      Distribute to 10.1.7.21 via IP routing   (   viparoute)  -
  ;-------------------------------------------------------------------
 ;;VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa
   VIPAROUTE      DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa       6

 ENDVIPADYNAMIC
```

1 SYSPLEXMONITOR DelayJoin keeps the stack from joining the sysplex until after OMPROUTE has completely initialized so that routing can be in full effect.

2 SYSPLEXROUTING enables sysplex distribution.

3 DYNAMICXCF assigns this stack to the XCF subnet and instructs the stack to participate in the sysplex

4 VIPADEFINE establishes the FTP VIPA address which clients use to access FTP

5 VIPADISTRIBUTE sets how to distribute connections to the FTP servers

6 VIPAROUTE is used in order to offload XCF by redirecting distributed traffic away from the XCF interfaces and toward the other available IP interfaces which are more efficient and usually faster.

The statements necessary to enable sysplex distribution in TCPIPB on LPAR SC31 are shown in Example 3-27.

Example 3-27 Sysplex enablement for TCPIP on SC31

```
GLOBALCONFIG
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   SYSPLEXROUTING
   DYNAMICXCF 10.1.7.21 255.255.255.0 8
;
VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.25   ;FTP
   VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                         10.1.8.25    PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using ROUNDROBIN         -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.21  ;FTP General
   VIPADISTRIBUTE DEFINE DISTMETHOD ROUNDROBIN
                         10.1.8.21   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using BASEWLM            -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.22  ;FTP Admin
   VIPADISTRIBUTE DEFINE DISTMETHOD BASEWLM
                         10.1.8.22   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;  Set up Sysplex Distribution for FTP using SERVERWLM          -
  ;-------------------------------------------------------------------
   VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.23  ;FTP Payroll
   VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
                         10.1.8.23   PORT 20 21
                  DESTIP 10.1.7.11
                         10.1.7.21

  ;-------------------------------------------------------------------
  ;      Distribute to 10.1.7.11 via IP routing   (   viparoute)  -
  ;      Distribute to 10.1.7.21 via normal XCF   (no viparoute)  -
  ;-------------------------------------------------------------------
   VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa
 ;;VIPAROUTE      DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa

 ENDVIPADYNAMIC
```

Customize a second FTP server started task procedure

This second started task is for our second FTP server. The procedure can be modeled after our first server started task. We used the same procedure name and same procedure library for both servers and used system symbolics to provide unique names for the FTP configuration profile data sets. The catalogued procedure we used to start both FTP servers is shown in Example 3-9 on page 99.

Customize a second FTP server configuration data set

The FTP.DATA for both servers should be identical. The FTP.DATA file we used for both FTP servers is discussed in 3.2, “Basic FTP without security” on page 84, under the heading “Create the FTP.DATA for the server” on page 98.

Customize an OMPROUTE started task to support the second stack

This second started task is for our second OMPROUTE. It can be modeled after our first started task. We used the same procedure name and same procedure library for both servers and used system symbolics to provide unique names for the OMPROUTE configuration data sets. The omproute started task we used is exactly the same as the started task discussed in “Customize an OMPROUTE started task to support the second stack” on page 59.

Customize an OMPROUTE configuration for the second OMPROUTE

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

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

The omproute configuration we used is exactly the same as the configuration discussed in “Customize an OMPROUTE configuration for the second OMPROUTE” on page 59.

3.3.3 Activation and verification of FTP servers within sysplex

Issue the MVS START command on both systems:

```
S TCPIPB
```

If FTP is autologged in both stacks, then FTP will automatically start and this step can be skipped. Otherwise, issue the MVS START command for the FTPDB job on both systems:

```
S FTPDB
```

All of the verification tasks previously discussed in 3.2, “Basic FTP without security” on page 84 can be used to verify the FTPD server is running correctly in the sysplex. Additionally, a number of NETSTAT displays can be used to show the status of Dynamic and Distributed VIPA connections. The format of the system NETSTAT command is:

```
D TCPIP,TCPIPB,N,command,option
```

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

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

- Use NETSTAT VCRT to show dynamic VIPA connection routing table.
- Use NETSTAT VDPT to show dynamic VIPA destination port table.
- Use NETSTAT VIPADCFG to show current dynamic VIPA configuration.
- Use NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE.

Use NETSTAT VCRT to show dynamic VIPA connection routing table

VCRT displays the dynamic VIPA connection routing table information. For each table entry that represents an established dynamic VIPA connection or an affinity created by the passive-mode FTP, the DETAIL suboption additionally displays the policy rule, action information, and routing information. For each entry that represents an affinity created by the TIMEDAFFINITY parameter on the VIPADISTRIBUTE profile statement, it displays the preceding information plus the affinity-related information.

Example 3-28 shows the connections at the time the VCRT command was issued. Notice that the distributing stack knows about all of the connections because it is managing them.

Example 3-28 NETSTAT VCRT on system SC30, the distributing stack

```
RO SC30,D TCPIP,TCPIPB,N,VCRT
   . . .
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.1.8.21..21
     SOURCE:  10.1.1.20..1029
     DESTXCF: 10.1.7.11
   DEST:      10.1.8.21..21
     SOURCE:  10.1.1.30..1026
     DESTXCF: 10.1.7.21
   2 OF 2 RECORDS DISPLAYED
   END OF THE REPORT
```

Notice that the non-distributing stack shows only the connections that have been distributed to it, as seen in Example 3-29.

Example 3-29 NETSTAT VCRT on system SC31, the non-distributing stack

```
RO SC31,D TCPIP,TCPIPB,N,VCRT
   . . .
   DYNAMIC VIPA CONNECTION ROUTING TABLE:
   DEST:      10.1.8.21..21
     SOURCE:  10.1.1.30..1026
     DESTXCF: 10.1.7.21
   1 OF 1 RECORDS DISPLAYED
   END OF THE REPORT
```

Use NETSTAT VDPT to show dynamic VIPA destination port table

VDPT displays the dynamic VIPA destination port table information. If the DETAIL suboption is specified, the output contains policy action information, target responsiveness values, and a WQ value (on a separate line).

Example 3-30 shows the port table entries at the time of issuing the VDPT command. SC30 is currently the distributor, so it shows the ports being distributed and whether there is a ready listener on the port.

Note: The TOTALCONN field indicates the total number of connections there have been since the distribution started for the port. It does *not* represent the current number of connections.

Example 3-30 NETSTAT VDPT on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VDPT
   . . .
   DYNAMIC VIPA DESTINATION PORT TABLE:
   DEST:        10.1.8.25..21
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.25..21
     DESTXCF:   10.1.7.21
     TOTALCONN: 0000000000  RDY: 000  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.21..21
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000002  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.1.8.21..21
     DESTXCF:   10.1.7.21
     TOTALCONN: 0000000001  RDY: 001  WLM: 01  TSR: 100
     FLG: ROUNDROBIN
   DEST:        10.1.8.22..21
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.22..21
     DESTXCF:   10.1.7.21
     TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
     FLG: BASEWLM
   DEST:        10.1.8.23..21
     DESTXCF:   10.1.7.11
     TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
     FLG: SERVERWLM
   DEST:        10.1.8.23..21
     DESTXCF:   10.1.7.21
     TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
     FLG: SERVERWLM
   10 OF 10 RECORDS DISPLAYED
   END OF THE REPORT
```

SC31 is not a distributor at the moment; therefore, it shows no information, as seen in Example 3-31.

Example 3-31 NETSTAT VDPT on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VDPT
   . . .
   DYNAMIC VIPA DESTINATION PORT TABLE:
   0 OF 0 RECORDS DISPLAYED
   END OF THE REPORT
```

Archived

Use NETSTAT VIPADCFG to show current dynamic VIPA configuration

VIPADCFG displays the current dynamic VIPA configuration information from the perspective of the stack on which the command is entered. The primary distributor shows the VIPA DEFINE, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 3-32.

Example 3-32 NETSTAT VIPADCFG on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADCFG
   . . .
   DYNAMIC VIPA INFORMATION:
     VIPA DEFINE:
       IPADDR/PREFIXLEN: 10.1.8.25/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.1.8.21/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.1.8.22/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
       IPADDR/PREFIXLEN: 10.1.8.23/24
         MOVEABLE: IMMEDIATE  SRVMGR: NO
     VIPA RANGE:
       IPADDR/PREFIXLEN: 10.1.9.0/24
         MOVEABLE: NONDISR
     VIPA DISTRIBUTE:
       DEST:        10.1.8.25..21
         DESTXCF:   10.1.7.11
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.25..21
         DESTXCF:   10.1.7.21
           SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.21..21
         DESTXCF:   10.1.7.11
           SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
       DEST:        10.1.8.21..21
         DESTXCF:   10.1.7.21
           SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
       DEST:        10.1.8.22..21
         DESTXCF:   10.1.7.11
           SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.22..21
         DESTXCF:   10.1.7.21
           SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
       DEST:        10.1.8.23..21
         DESTXCF:   10.1.7.11
           SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
       DEST:        10.1.8.23..21
         DESTXCF:   10.1.7.21
           SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
     VIPA ROUTE:
       DESTXCF:     10.1.7.21
         TARGETIP:  10.1.1.20
END OF THE REPORT
```

The backup stack shows the VIPA BACKUP, RANGE, DISTRIBUTE, and ROUTE sections, as shown in Example 3-33.

Example 3-33 NETSTAT VIPADCFG on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADCFG
   . . .
   DYNAMIC VIPA INFORMATION:
```

```
    VIPA BACKUP:
      IPADDR/PREFIXLEN: 10.1.8.25
        RANK: 200  MOVEABLE:           SRVMGR:
      IPADDR/PREFIXLEN: 10.1.8.21
        RANK: 200  MOVEABLE:           SRVMGR:
      IPADDR/PREFIXLEN: 10.1.8.22
        RANK: 200  MOVEABLE:           SRVMGR:
      IPADDR/PREFIXLEN: 10.1.8.23
        RANK: 200  MOVEABLE:           SRVMGR:
    VIPA RANGE:
      IPADDR/PREFIXLEN: 10.1.9.0/24
        MOVEABLE: NONDISR
    VIPA DISTRIBUTE:
      DEST:        10.1.8.25..21
        DESTXCF:   10.1.7.11
          SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
      DEST:        10.1.8.25..21
        DESTXCF:   10.1.7.21
          SYSPT:   YES  TIMAFF: NO    FLG: BASEWLM
      DEST:        10.1.8.21..21
        DESTXCF:   10.1.7.11
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.1.8.21..21
        DESTXCF:   10.1.7.21
          SYSPT:   NO   TIMAFF: NO    FLG: ROUNDROBIN
      DEST:        10.1.8.22..21
        DESTXCF:   10.1.7.11
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.1.8.22..21
        DESTXCF:   10.1.7.21
          SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM
      DEST:        10.1.8.23..21
        DESTXCF:   10.1.7.11
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      DEST:        10.1.8.23..21
        DESTXCF:   10.1.7.21
          SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    VIPA ROUTE:
      DESTXCF:     10.1.7.11
        TARGETIP:  10.1.1.10
  END OF THE REPORT
```

Use NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE

VIPADYN displays the current dynamic VIPA and VIPAROUTE information from the perspective of the stack on which the command is entered. There are two suboptions available to filter the output:

- DVIPA - Displays the current dynamic VIPA information only
- VIPAROUTE - Displays the current VIPAROUTE information only

Example 3-34 shows SC30 information.

Example 3-34 NETSTAT VIPADYN on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN
   . . .
   DYNAMIC VIPA:
     IPADDR/PREFIXLEN: 10.1.8.25/24
       STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
       ACTTIME: 09/23/2007 11:35:40
```

```
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 09/23/2007 11:35:40
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 09/23/2007 11:35:40
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 09/23/2007 11:35:40
  VIPA ROUTE:
    DESTXCF: 10.1.7.21
      TARGETIP: 10.1.1.20
      RTSTATUS: ACTIVE
  5 OF 5 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 3-35 shows the same information for SC30.

Example 3-35 NETSTAT VIPADYN,DVIPA on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
  . . .
  DYNAMIC VIPA:
  IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 09/23/2007 11:35:40
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 09/23/2007 11:35:40
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 09/23/2007 11:35:40
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
      ACTTIME: 09/23/2007 11:35:40
  4 OF 4 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 3-36 shows SC30 Netstat output for VIPADYN,VIPAROUTE, with filters for viparoute only.

Example 3-36 NETSTAT VIPADYN,VIPAROUTE on system SC30

```
RO SC30,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
  . . .
  VIPA ROUTE:
    DESTXCF: 10.1.7.21
      TARGETIP: 10.1.1.20
      RTSTATUS: ACTIVE
  1 OF 1 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 3-37 shows SC31Netstat output for VIPADYN ftp and FTP dvipa addresses.

Example 3-37 NETSTAT VIPADYN on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN
  . . .
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.25/24
```

```
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
  VIPA ROUTE:
    DESTXCF: 10.1.7.11
      TARGETIP: 10.1.1.10
      RTSTATUS: ACTIVE
  5 OF 5 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 3-38 shows SC31 Netstat output for VIPADYN,DVIPA: with filters for dvipa only.

Example 3-38 NETSTAT VIPADYN,DVIPA on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,DVIPA
  . . .
  DYNAMIC VIPA:
    IPADDR/PREFIXLEN: 10.1.8.25/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.21/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.22/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
    IPADDR/PREFIXLEN: 10.1.8.23/24
      STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
  4 OF 4 RECORDS DISPLAYED
  END OF THE REPORT
```

Example 3-39 shows SC31 Netstat output for VIPADYN,VIPAROUTE, with filters for viparoute only.

Example 3-39 NETSTAT VIPADYN,VIPAROUTE on system SC31

```
RO SC31,D TCPIP,TCPIPB,N,VIPADYN,VIPAROUTE
  . . .
  VIPA ROUTE:
    DESTXCF: 10.1.7.11
      TARGETIP: 10.1.1.10
      RTSTATUS: ACTIVE
  1 OF 1 RECORDS DISPLAYED
  END OF THE REPORT
```

3.4 FTP client via batch

In this section we discuss running the FTP client as a batch job. The following topics are covered:

- Description of FTP client via batch
- Configuration of FTP client via batch
- Activation and verification of FTP client batch job

Archived

3.4.1 Description of FTP client via batch

In addition to the dependencies discussed in 3.2, “Basic FTP without security” on page 84, an FTP batch job requires JCL containing the FTP commands to be executed.

Submitting batch FTP jobs allows non-interactive FTP sessions that run in the background.

Batch FTP does not handle transient failures. For example, if there is a network problem, the FTP batch job will simply fail. If the network problem is resolved, the batch job must be resubmitted to complete the file transfer.

3.4.2 Configuration of FTP client via batch

Example 3-40 shows a sample batch job.

Example 3-40 Sample FTP client batch job

```
//BATFTP   JOB MSGCLASS=X,NOTIFY=&SYSUID
//FTPCLNT  EXEC PGM=FTP,PARM='(TIMEOUT 15'                1
//NETRC    DD DSN=CS07.NETRC,DISP=SHR                     2
//SYSFTPD  DD DSN=TCPIPB.TCPPARMS(FTPCB),DISP=SHR         3
//OUTPUT   DD *
//SYSPRINT DD *
//INPUT    DD *
10.1.1.10                                                 4
LS                                                        5
QUIT                                                      6
/*
```

The following items explain Example 3-40:

1 Client parameters can be passed to the client by way of the PARM= field. Refer to *z/OS Communications Server: IP User’s Guide and Commands*, SC31-8780 for a complete list of parameters and their syntax.

2 The NETRC dataset can be used to provide the user ID and password to the FTP client program, which in turn, forwards the user ID and password to the server when prompted by the server. We used the NETRC dataset from which to retrieve the user ID and password so they would not be exposed in the input command stream. If you do not include the NETRC dataset in the JCL, both the user ID and the password must be included in the INPUT stream immediately after the target device specification.

3 The SYSFTPD statement points to the FTP.DATA client configuration file. Refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for a complete list of statements and their settings.

4 The first entry in the INPUT command file is the IP address or DNS name of the target server to which to connect. Alternatively, the target can be specified in the PARM= field on the EXEC statement, and if it is, it should not be specified in the IINPUT command file. Here is a sample where the target is specified in the PARM= field:

```
//FTPCLNT EXEC PGM=FTP,PARM=’10.1.1.10 (TIMEOUT 15’
```

5 After the target specification, you can include any FTP subcommand in the INPUT stream. See *z/OS Communications Server: IP User’s Guide and Commands*, SC31-8780 for a complete list of FTP subcommands that can be specified.

6 The last subcommand should always be either CLOSE or QUIT

Example 3-41 shows a sample NETRC dataset.

Example 3-41 Sample NETRC dataset

```
********************************** Top of Data **********************
  1        2                          3     4          5      6
machine 127.0.0.1                    login CS07     password CS07
machine wtsc30.itso.ibm.com          login CS07     password CS07
machine wtsc31.itso.ibm.com          login CS07     password CS07
machine wtsc32.itso.ibm.com          login CS07     password CS07
machine wtsc33.itso.ibm.com          login CS07     password CS07
machine 10.1.1.10                    login CS07     password CS07
machine 10.1.1.20                    login CS07     password CS07
machine 10.1.1.30                    login CS07     password CS07
machine 10.1.1.40                    login CS07     password CS07
******************************** Bottom of Data *******************
```

The following items explain Example 3-41:

1, 3, 5 The words *machine*, *login*, and *password* are keywords and must be spelled as seen here. They must also be in the same order as seen here. However, they are *not* case sensitive. If you do not provide the *login* keyword and *user ID value* on a specific machine record, the client expects to find the user ID in the INPUT stream. If you do not provide the password value after the *password* keyword on a specific machine record, the client expects to find the password in the INPUT stream (that would be a bad choice to make).

2 The remote device can be specified with either its IP address or its DNS name, or both, as shown.

4 Your user ID on the remote system must follow the *login* keyword. It is case sensitive only if RACF requires case sensitive user IDs.

6 Your password on the remote system must follow the *password* keyword. It is case sensitive only if RACF requires case sensitive passwords.

3.4.3 Activation and verification of FTP client batch job

Submit the batch job for execution and check its output log. Example 3-42 shows the client output log from our batch job.

Example 3-42 Sample output log for FTP batch job

```
1 EZA1736I FTP (TIMEOUT 15 TCP TCPIPB
2 EZY2640I Using dd:SYSFTPD=TCPIPB.TCPPARMS(FTPCB) for local site configuration
  EZA1466I FTP: using TCPIPB
  EZA1456I Connect to ?
  EZA1736I 10.1.1.10
3 EZA1554I Connecting to:   10.1.1.10 port: 21.
4 220-FTPDB1 IBM FTP CS at WTSC30.ITSO.IBM.COM, 19:41:44 on 2007-09-24.
  220 Connection will close if idle for more than 5 minutes.
5 EZA1701I >>> USER CS07
  331 Send password please.
  EZA1701I >>> PASS
6 230 CS07 is logged on.  Working directory is "CS07.".
7 EZA1460I Command:
8 EZA1736I LS
  EZA1701I >>> PORT 10,1,1,10,4,22
  200 Port request OK.
  EZA1701I >>> NLST
```

```
  125 List started OK.
  EZA2284I BRODCAST
  EZA2284I HFS
  EZA2284I JCL.CNTL
  EZA2284I NETRC
  250 List completed successfully.
  EZA1460I Command:
9 EZA1736I QUIT
```

1 The FTP client program echoes the PARM= field contents

2 The FTP.DATA file that the resolver located is listed

3 The FTP client indicates what IP address and port it is connecting to

4 All server reply messages start with a 3-digit number. Server replies are documented in *z/OS Communications Server: IP and SNA Codes*, SC31-8791.

5 The FTP client program has retrieved the user ID (CS07) and password from the NETRC dataset and sends them to the server with the USER and PASS subcommands, respectively.

6 The server replies with message 230, indicating that it has accepted the user ID and password and has successfully logged the user in. It has set the default working directory to a high level qualifier that is equal to the user's ID (CS07.)

7 The client FTP program prompts the user for the next subcommand. The commands are being read from the INPUT stream by the client program.

8 The next command from the INPUT stream is LS, and is sent to the server by the client program. The server replies with the 125 message, the requested info, and the 250 message.

9 The last command should always be QUIT.

3.5 Miscellaneous configurations of FTP

There are some other useful functions of FTP that we do not cover in this book. However, you should be aware of the capabilities.

The following topics are discussed:

- FTP client via the FTP client API - REXX support
- A single generic FTP server in a multiple stack z/OS image
- FTP network management interface with SMF

3.5.1 FTP client via the FTP client API - REXX support

An application using the FTP client API must be written in COBOL, C, Assembler, PL/1, or REXX. *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787, lists additional requirements of the application. In this section we discuss the use of the FTP client API REXX support.

The FTP client API allows any application to directly invoke the functionality of the FTP client running on z/OS with significantly improved automation capabilities for file transfer operations. REXX skills are becoming more prevalent due to the inherent ease in programming that it provides. As such, many people can quickly and easily develop REXX programs to accomplish complex tasks.

The REXX FTP Client API is supported in the following environments:

- Non-authorized TSO exec
- Authorized TSO exec
- Batch environment
- ISPF
- UNIX environment under UNIX System shell scripts

As with an FTP batch job, an application that uses the FTP client API must handle transient errors or the FTP transfer will fail.

A REXX FTP client requires a stub routine to translate between the string format used within REXX programs and the text/binary format used by the underlying callable FTP Client API.

REXX FTP Client API functions share a common return code format.

3.5.2 A single generic FTP server in a multiple stack z/OS image

In this section we discuss starting a single FTP server to handle connections through multiple stacks on the same LPAR. The multiple stacks per z/OS image is referred to as a CINET environment.

Dependencies of a generic FTP server

Multiple stacks on the LPAR must be configured and active.

Advantages of a generic FTP server

If you need two or more stacks configured differently but only one FTP server is required, then the FTP server can be set up as a generic listener. This configuration enables the FTP server to listen for connections on all stacks. For example, you could have one stack with a firewall and policies active which processes connections from an external network, and one stack without a firewall and policies which processes connections from the internal network. If one single FTP configuration is capable of meeting the needs of all stacks, then this configuration has an additional advantage: only one FTP instance needs to be configured.

Considerations for using a generic FTP server

Additional stacks require additional resources in terms of CPU load and memory usage. Additional system definitions are required to accommodate multiple stacks per system image (known as a CINET environment). Running multiple stacks per system image is not recommended. The requirement for multiple stacks is best accomplished using multiple LPARs and sysplexes.

3.5.3 FTP network management interface with SMF

System Management Facility (SMF) is a component of z/OS. It is used to help monitor z/OS systems by capturing and recording events that occur. After these events are recorded, reports may be generated either by user-written programs that format the data into a readable format or by real time network monitors for display. These recorded events are referred to as SMF Records, and are stored in one of two places:

- SMF Datasets - VSAM datasets, typically named SYS1.MANx, SYS1.MANy, and so on
- z/OS Dataspaces

SMF datasets must have one dataset actively recording data.

The SMF records for FTP are as follows:

- Three SMF records provide session information:
  - FTP client login failure
  - FTP client session
  - FTP server session
- Five records provide more detailed information:
  - FTP client transfer completion record
  - FTP server transfer completion record
  - FTP server login failure
  - FTP client transfer initialization
  - FTP server transfer initialization

Advantages

The SMF data can be accessed in two ways:

- Through normal batch processing of SMF data
- Through a real-time Network Management interface

For normal SMF data you need to configure ftp.data as well as profile.tcpip to specify that SMF records are to be created when certain events occur.

- Code the SMF statement **SMFCONFIG TYPE119 FTPCLIENT** in the profile.tcpip file;
- Code the SMF statements in the FTP server's ftp.data file as shown in Example 3-43:

Example 3-43 Definitions in the ftp.data file.

```
SMF      STD       ; SMF type 118

SMF      TYPE119   ; SMF type 119
```

The real-time network management data is enabled by including a **NETMONITOR SMFSERVICE** statement in profile.tcpip.

3.6 Problem determination for FTP

In case of a server problem, first check the console and syslogd for error messages. If no problem can be identified from messages, enable the FTP TRACE by specifying the keyword TRACE on the first line of the FTPD catalogued procedure on the PARMS parameter.

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

3.7 Additional information sources for FTP

For additional information, refer to:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- FTP is defined by RFC 959

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Archived

Archived

4

Simple Network Management Protocol

Simple Network Management Protocol (SNMP) enables monitoring and management of the devices and computers participating in a TCP/IP network. This chapter focuses on the SNMP services that are available in the z/OS Communications Server, and complements the product publications with practical implementation scenarios that can be useful in your environment.

A Management Information Base (MIB) provides the core definition of all network-managed resources. Managed data is defined in IETF standards, and in proprietary instances the data are called *MIB objects* or *MIB variables*. MIB-II is the recommended Internet standard format for managed information. Its current specification can be found in RFC 1213, as well as RFC2011 through RFC2013 (Management Information Base for Network Management of TCP/IP-based internetworks: MIB-II).

This chapter discusses the following SNMP topics.

Section	Topic
4.1, “Conceptual overview of SNMP” on page 128	The basic concepts of SNMP network management.
4.2, “z/OS SNMP agent” on page 131	SNMP Agent configuration setup examples and verification procedures.
4.3, “z/OS SNMP subagents” on page 142	SNMP Subagent configuration setup examples and verification procedures.
4.4, “z/OS SNMP client command” on page 148	SNMP client command configuration setup examples and usage procedures.
4.5, “Problem determination for the SNMP facilities” on page 156	Problem determination tools and commands for z/OS SNMP facilities.
4.6, “Additional information sources for SNMP” on page 157	References to additional documentation for z/OS SNMP facilities.

© Copyright IBM Corp. 2008. All rights reserved.

4.1 Conceptual overview of SNMP

As illustrated in Figure 4-1, SNMP is one of the standard applications provided with the z/OS Communications Server. Both the SNMP agent and the SNMP command send and receive message packets through z/OS UNIX Sockets, using the Assembler Callable API and C-Sockets. They make use of the Logical File System to pass the packets in and out of the Physical File System on to the layers of the stack.

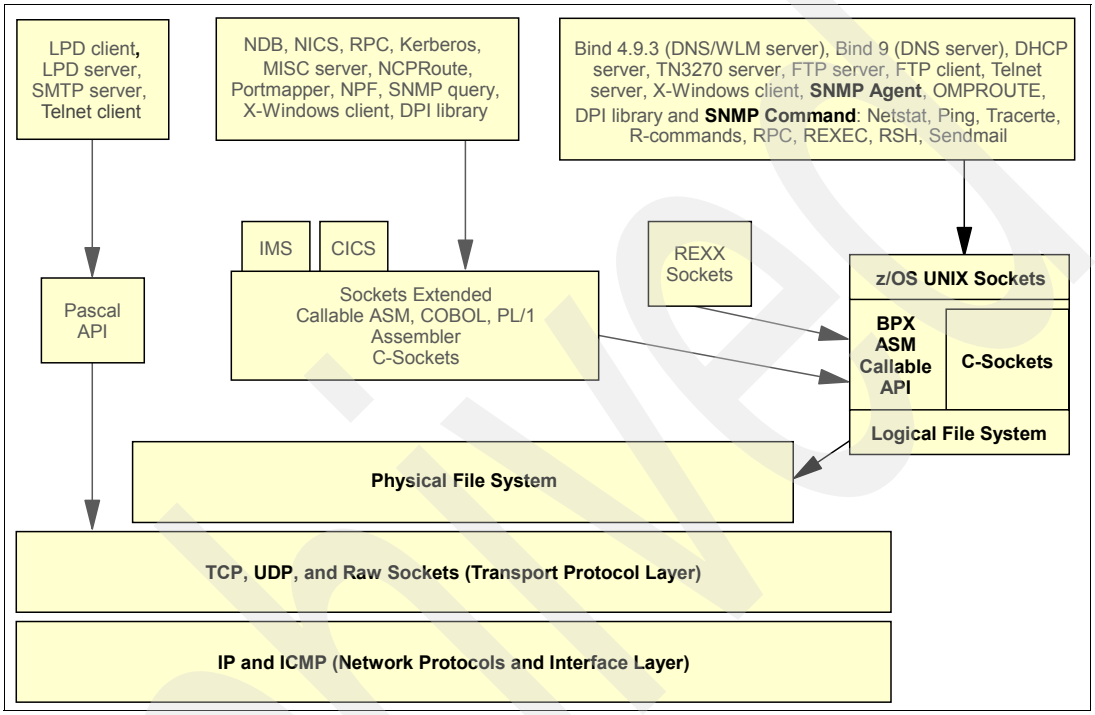


Figure 4-1 z/OS SNMP management services

As IP networks have grown in size and complexity, so has the need for managing them. Management packages are becoming more sophisticated and are now requiring the availability of an SNMP agent. The z/OS platform provides the robust support and high availability that is required by these packages.

SNMP is probably the most widely used application for managing elements in a TCP/IP network. For most vendors, it has become an integral part of their packaged network management offerings. The versatility of the z/OS platform enables it to support both SNMP managers and SNMP agents for collecting, monitoring, and reporting network statistics. A number of the applications shipped with the z/OS Communications Server have their own subagent that can be enabled to register with an SNMP agent on the same system image.

This section discusses the following topics:

- What is SNMP?
- How does SNMP work?
- How can SNMP be applied?

4.1.1 What is SNMP?

SNMP is an Internet standard protocol. Its current specification can be found in RFC 1157 Simple Network Management Protocol (SNMP). SNMP makes use of managers, agents, and subagents. The subagents are closely associated with managed elements that are able to

access specific device status information. The agents provide interfaces on behalf of subagents to management packages that want to retrieve that information.

The SNMP framework enables network administrators to address various networking management related issues. Because these operations are done using the SNMP protocol, a network administrator can reside anywhere in the IP network, and no longer has to log into the target systems to maintain network nodes.

The framework of version 2 of the Simple Network Management Protocol (SNMPv2) was published in April 1993 and consists of 12 RFCs—the first being RFC 1441 (which is an introduction). In August 1993, all 12 RFCs became proposed standards with the status *elective*. SNMPv3 is described in RFCs 2570 through 2573 and RFCs 3411 through 3415. SNMPv3 is an extension to the existing SNMP architecture.

The view-based access control model supported in SNMPv3 allows granular access control for MIB objects with either the user-based or community-based security models.

SNMPv3 also enables dynamic changes to the SNMP agent configuration. The SNMPv3 architecture is modularized so that portions of it can be enhanced over time without requiring the entire architecture to be replaced.

4.1.2 How does SNMP work?

A Network Management Station (NMS) requests an item of information from the agent to which the NMS has an IP connection. The agent forwards that request to an appropriate subagent that has registered itself as supporting that type of information. The subagent prepares a response and sends the information back to the agent. The agent forwards the response back to the NMS. The NMS is not aware of how the agent acquires the information.

The SNMP protocol implementation is illustrated in Figure 4-2.

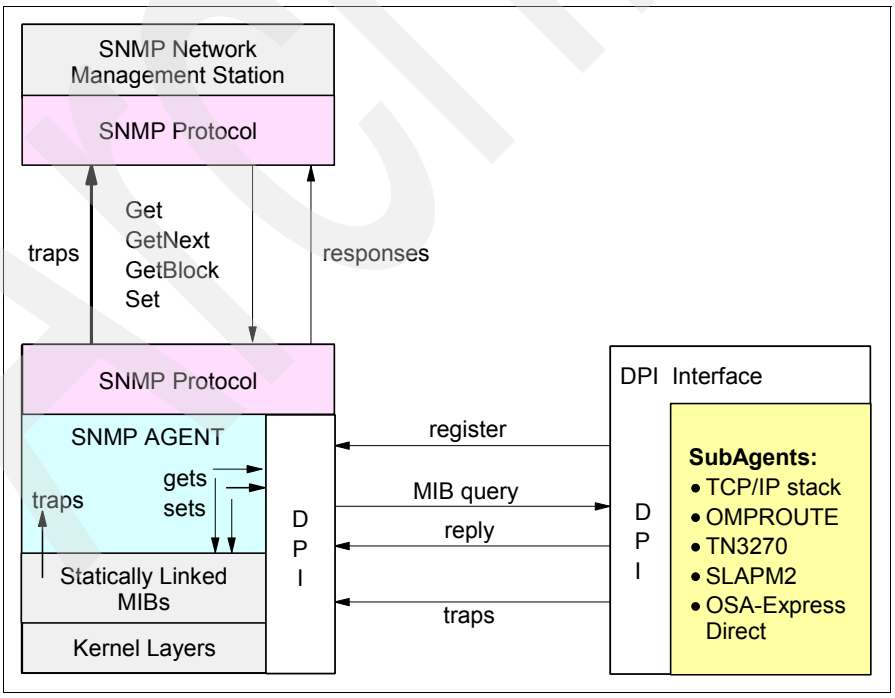


Figure 4-2 SNMP agent/subagent relationship to a Network Management Station (manager)

The SNMP Distributed Programming Interface (DPI®) allows a process to register the existence of an MIB variable with the SNMP agent. When requests for the variable are received by the SNMP agent, it passes the query on to the process acting as a subagent. This subagent then returns an appropriate answer to the SNMP agent. The SNMP agent packages an SNMP response packet and sends the answer back to the remote network management station that initiated the request. The network management stations have no knowledge that the SNMP agent calls on other processes to obtain an answer.

- The SNMP manager (NMS) communicates with the SNMP agent via the SNMP protocol.
- The SNMP agent communicates with the Management Information Base that represents the items of information available.
- An SNMP subagent, running as a separate process, can set up a connection with the agent. The subagent has an option to communicate with the SNMP agent through UDP or TCP sockets, or even through other mechanisms.
- Once the connection is established, the subagent registers one or more of its MIBs with the SNMP agent. Multiple subagents can register with the agent concurrently. In our environment, those subagents could be from TN3270, OMPROUTE, the TCP/IP stack, Infoprint Server, and others.
- The SNMP agent receives request packets from the SNMP manager (NMS). If the packet contains a request for an object in an MIB registered by a subagent, it sends a corresponding request in a Distributed Programming Interface (DPI) packet to the subagent.
- The SNMP agent then encodes a reply into an SNMP packet and sends it back to the requesting SNMP manager.

To understand the relationship between the NMS, the agent, and the subagent, an understanding of the Distributed Programming Interface (DPI) is necessary. This is a special-purpose programming interface that you can use if you want to implement Management Information Base (MIB) variables. In an z/OS Communications Server SNMP environment the MIB variables are defined in the MIBS.DATA data set. If you want to add, replace, or delete MIB variables, you can develop an SNMP subagent program that uses the DPI programming interface to interact with the SNMP agent address space (OSNMPD) to perform such functions. For details on using the DPI interface see *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787.

Some of the subagents and their specific MIBs delivered with the z/OS Communications Server are:

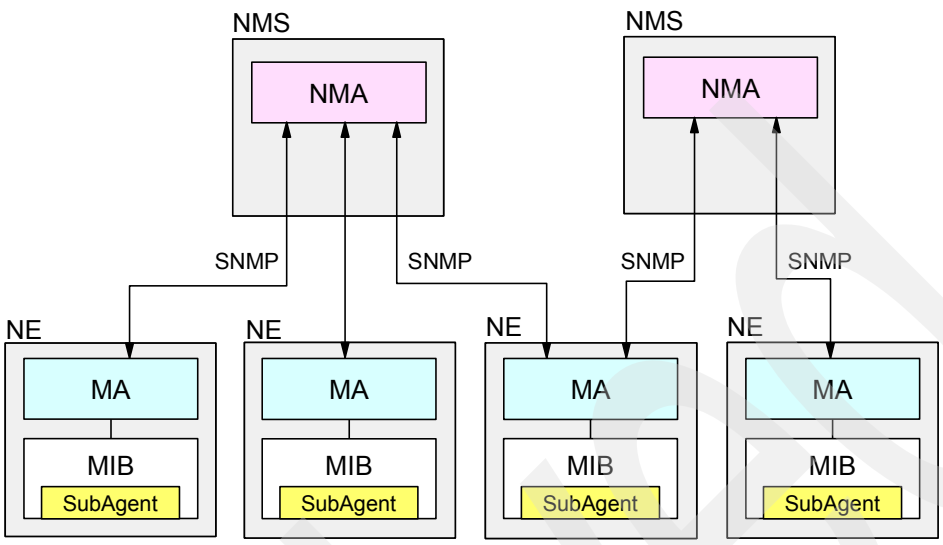
- TCP/IP stack subagent with /usr/lpp/tcpip/samples/mvstcpip.mi2
- OMPROUTE subagent
- TN3270E Telnet server subagent with /usr/lpp/tcpip/samples/mvstn3270.mi2
- NSLAPM2 V2 subagent for policy agent with usr/lpp/tcpip/samples/slapm2.mi2
- OSA-Express Direct subagent supports data for OSA-Express features

The SNMP DPI V2.0 protocol specified in RFC1592 provides the ability to connect agents via AF_UNIX or AF_INET sockets. The list of SNMP RFCs is large, and it is out of the scope of this book to discuss them. For complete details on the standards and how they relate to each other, refer to *TCP/IP Tutorial and Technical Overview*, GG24-3376.

4.1.3 How can SNMP be applied?

The diagram in Figure 4-3 on page 131 illustrates the generic implementation of SNMP in a common industry standard design. It shows the management component layers and the roles they play in network management. The IBM z/OS implementation does not use all of the

terminology defined in the diagram; however, the diagram and its following explanation will assist in understanding how SNMP is generally implemented.



Acronym	Term	Example
NMS	Network Management Station	z/OS system (APPN NN)
NMA	Network Management Application	NetView or osnmp UNIX command
NE	Network Element	z/OS system (APPN EN)
MA	Management Agent	z/OS SNMPD agent daemon
MIB	Management Information Base	MIB-II or SubAgent specific MIB

Figure 4-3 SNMP Network Management roles

A host platform that runs a management package is a Network Management Station, and the package is the Network Management Application (or manager). The manager communicates with an agent using the SNMP protocol. The agent communicates with subagents to retrieve information that satisfies requests from the manager. The protocol used between an agent and a subagent can be DPI/SMUX/AgentX or any proprietary protocol. The z/OS Agent uses the DPI interface. A host platform that runs the agent and possibly the subagent is the network element.

These two SNMP implementations are discussed in this chapter:

- Simple Network Management Protocol (SNMP) agent and subagent in the MVS environment
- The z/OS UNIX snmp command as a manager requesting information from the agent

4.2 z/OS SNMP agent

A common implementation of SNMP is to set up the SNMP agent (OSNMPD) as a started task on the z/OS platform under the native MVS environment. We discuss the configuration of OSNMPD as an agent in the following topics:

- “Description of the z/OS SNMP agent” on page 132
- “Configuration of the z/OS SNMP agent” on page 133
- “Activation and verification of the z/OS SNMP agents” on page 141

4.2.1 Description of the z/OS SNMP agent

In the z/OS Communications Server, the **snmp** command provides SNMP network management from the z/OS UNIX shell. The Tivoli® NetView SNMP command provides network management from the NetView command line.

In the z/OS Communications Server, the SNMP agent is a z/OS UNIX application. It supports SNMPv1, SNMPv2c, and SNMPv3. SNMPv3 provides a network management framework that enables the use of user-based security in addition to, or instead of, the community-based security supported in SNMPv1 and SNMPv2c.

The z/OS SNMP agent and its relationship with the IP network is illustrated in Figure 4-4.

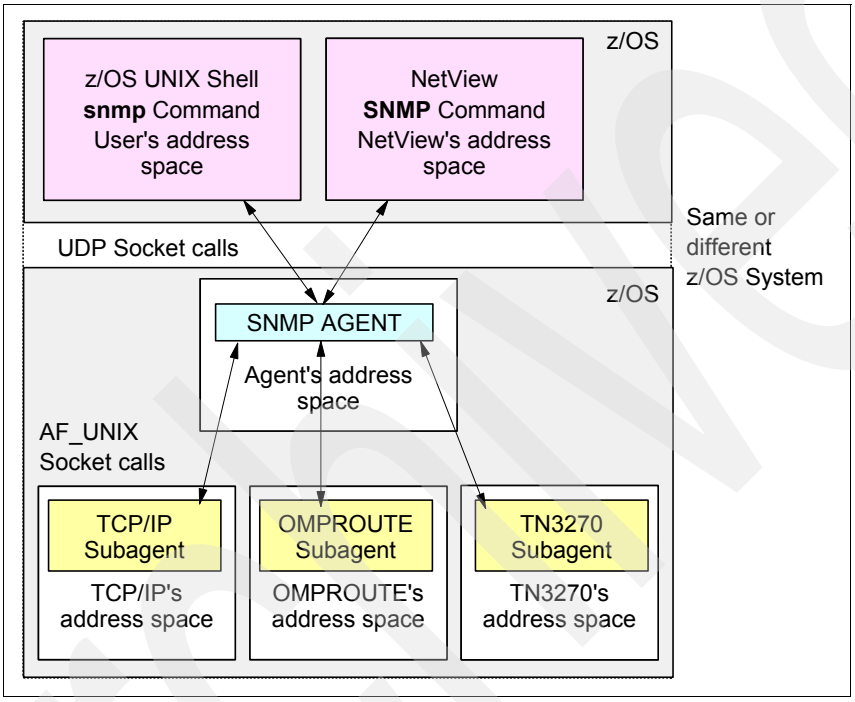


Figure 4-4 SNMP agent/subagent relationships

Dependencies of the z/OS SNMP agent

A subagent extends the set of MIB variables supported by an SNMP agent. Each subagent must be set up and configured independently. The z/OS Communications Server supports the following subagents:

- TCP/IP subagent with /usr/lpp/tcpip/samples/mvstcpip.mi2
- OMPROUTE subagent
- TN3270 Telnet subagent with /usr/lpp/tcpip/samples/mvstn3270.mi2
- Network SLAPM2 subagent with /usr/lpp/tcpip/samples/slapm2.mi2
- OSA-Express Direct subagent supports data for OSA-Express features

The TCP/IP stack must be active on the z/OS system on which the SNMP agent runs. The z/OS UNIX environment must be enabled and active. The agent is a z/OS UNIX application.

The trap forwarder task forwards traps from the SNMP agent to network management applications. It listens for traps on a port, typically 162, and forwards them to all configured managers. If you want to forward trap information to one or more managers, you must configure the trap forwarder. For setup and configuration details, see *z/OS Communications*

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

TN3270 SNMP subagent limitation

TN3270 SNMP subagent activation requires specification of a stack name to register with the agent. Without TCPIPJOBNAME, TN3270 blocks the subagent activation request.

The TN3270 SNMP subagent can only register with one agent, and each agent can support only one TN3270 subagent. In addition to the required affinity, you must be careful to plan for one agent per TN3270 subagent, including the TN3270 subagent that might be running within the TCP/IP stack's address space. If multiple TN3270 SNMP subagents initialize to the same agent, the agent forwards all data requests to the first subagent that connected, and all other initializations are queued. If the first subagent ends, the next subagent in the queue then receives all data requests.

Considerations for using the z/OS SNMP agent

Migrating from SNMPV2 to SNMPv3 takes planning and could be a complex task. For information about migrating z/OS SNMP configuration files from SNMPv1 and SNMPv2c to SNMPv3, refer to the SNMP chapter in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

4.2.2 Configuration of the z/OS SNMP agent

Depending on the network management requirements of your organization, you may have to implement certain features of the SNMP agent on your z/OS platform. There are common steps to set up the SNMP agent regardless of how it is going to be used or what role it plays. We show you how to set up the basic SNMP agent in this section. The subagents are set up independently. Each has its own set of definitions and configuration file. The diagram in Figure 4-4 on page 132 is referred to in the following implementation tasks for the z/OS SNMP agent:

- Update the TCPIP profile configuration data set for the agent
- Update the TCPIP profile configuration data set for the query engine
- Update RACF to define the SNMP agent started task
- Customize the SNMP agent procedure JCL
- Create the environment variable file for the SNMP agent proc
- Create the MIB object configuration file for the SNMP agent
- Determine the type of security supported by the SNMP agent
- Create the community name file for the SNMP agent
- Create the trap destination file for the SNMP agent
- Configure the query engine started task (optional)
- Create the SNMPARMS file to be use by the Netview query engine
- Configure the trap forwarder started task and its files (optional)

Update the TCPIP profile configuration data set for the agent

The AUTOLOG and PORT statements should be updated to indicate the action and support that the stack needs to provide for the SNMP agent. AUTOLOG indicates whether the stack should initially start the SNMP started task. PORT provides a port reservation for the port number that the SNMP server listens on. The default is port 161. The AUTOLOG and PORT statements are shown in Example 4-1.

Example 4-1 Auto starting and port reservation for the SNMPD started task

```
AUTOLOG
   SNMPDB
```

```
ENDAUTOLOG

PORT
   161 UDP SNMPDB
```

Update the TCPIP profile configuration data set for the query engine

The SNMP agent uses port 162, by default, for sending traps to the managers specified in the SNMPTRAP.DEST or the SNMPD.CONF file. Port 162 should be reserved for the management application primarily responsible for trap processing. If you plan to use the query engine (one of the users of this is Tivoli NetView), then reserve the port as shown in Example 4-2.

Example 4-2 Auto starting and port reservation for the SNMPQE query engine

```
AUTOLOG
   SNMPQEB
ENDAUTOLOG

PORT
   162 UDP SNMPQEB
```

Update RACF to define the SNMP agent started task

Every started task must be assigned a user ID, and that user ID must be granted authority to access the required resources that support the started task. The started tasks that need to be considered here are:

- SNMPD, the agent's started task
- SNMPQE, the query engine
- TRAPFWD, the trap forwarder

The SNMP agent started task is used as an example. This discussion assumes RACF is the security subsystem being used. If another security product is used, refer to its manuals for equivalent set up instructions. Before SNMP can be started, security for the procedure name and its associated user ID must be defined. Review the sample file SEZAINST(EZARACF) that contains sample security statements for this effort.

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

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

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

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

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

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

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

1. Add the user to RACF:

   ```
   ADDUSER OSNMP
   ```

2. Permit the user ID:

   a. Create a BPX.SUPERUSER FACILITY class profile:

      ```
      RDEFINE FACILITY BPX.SUPERUSER
      ```

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

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

   c. Permit the user to the class:

      ```
      ALTUSER SNMP OMVS(UID(25) PROGRAM ('/bin/sh') HOME('/'))
      PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SNMP) ACCESS(READ)
      ```

      In this example, the user ID is SNMP and the UID is 25. The UID can be any nonzero number. UID 25 was used to match the well-known SNMP port number.

   d. Refresh the FACILITY class:

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

Customize the SNMP agent procedure JCL

A sample of the procedure is in hlq.SEZAINST(OSNMPDPR). Customize data set names to meet installation standards. There are some follow-on steps below that may require you to add one or more DD statements to this procedure. Store your updated procedure into your system proclib and make sure its name matches the name you put on the AUTOLOG and PORT statements in the TCPIP profile configuration data set.

Our SNMP agent started task procedure is shown in Example 4-3.

Example 4-3 SNMP agent Proc JCL

```
//SNMPDB PROC PARMS='-d 0',STDENV=SNMENV&SYSCLONE.              1
//SNMPDB EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/&PARMS')
//STDENV  DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV.)  2
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*
```

The following items explain Example 4-3:

- **1** The &SYSCLONE system variable is used to determine the name of the SNMPENV member that contains settings for environment variables used by SNMPD.
- **2** The STDENV DD statement should specify a PDS that has a variable record format (RECFM=V or VB).

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

RECFM=F must *not* be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in UNIX System Services. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Create the environment variable file for the SNMP agent proc

Example 4-4 shows the two STDENV files for our two agents. The SNMP agent requires information from the files that are indicated in the environment variable file. The PDSs must have a RECFM of VB in order to enable proper processing of the variable settings.

Example 4-4 SNMP agent STDENV files for SC30 and SC31

```
for SC30:
BROWSE    TCPIP.SC30.STDENV(SNMENV30) - 01.01 Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA                  1
RESOLVER_CONFIG=//'TCPIPA.TCPPARMS(DATAA30)'      2
OSNMPD_DATA=//'TCPIPA.TCPPARMS(SNMPD30)'          3
PW_SRC=//'TCPIPA.TCPPARMS(PWSRC30)'               4
SNMPTRAP_DEST=//'TCPIPA.TCPPARMS(TRPDST30)'       5

for SC31:
BROWSE    TCPIP.SC31.STDENV(SNMENV31) - 01.01 Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
OSNMPD_DATA=//'TCPIPB.TCPPARMS(SNMPD31)'
PW_SRC=//'TCPIPB.TCPPARMS(PWSRC31)'
SNMPTRAP_DEST=//'TCPIPB.TCPPARMS(TRPDST31)'
```

The following items explain Example 4-4:

- 1 _BPXK_SETIBMOPT_TRANSPORT establishes stack affinity to the indicated stack.
- 2 RESOLVER_CONFIG points to the TCPDATA resolver file.
- 3 OSNMPD_DATA points to the MIB object configuration file for the agent.
- 4 PW_SRC points to the community names file used by the agent
- 5 SNMPTRAP_DEST points to the trap destination file if needed by the agent

Create the MIB object configuration file for the SNMP agent

- Supplying this information permits you to customize the values of certain MIB objects for your specific installation. A sample, containing the MIB objects to be set, can be found in the z/OS UNIX file system as file */usr/lpp/tcpip/samples/osnmpd.data*. It can be copied to a PDS member if you want the SNMPD procedure to use it that way. Customize the settings to match your environment. Example 4-5 shows a portion of the osnmpd_data file.

Example 4-5 OSNMPD_DATA file TCPIPB.TCPPARMS(SNMPD31)

```
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "I&O Mainframe Network Support"                    1
  sysLocation "Datacenter Mainframe Operations"                  2
  sysName "SC31 - z/OS Communications Server"                    3
# Default value of sysObjectID is equivalent to ibmTcpIpMvs
```

```
#  in the ibmAgents subtree; this is the sysObjectID representing
#  IBM z/OS Communications Server
# Changing this value is not recommended, as it is intended to allow
#  network management applications to identify this agent as the
#  z/OS Communication Server SNMP agent.  The ability to change it
#  will be disabled in a subsequent release.
# sysObjectID     "1.3.6.1.4.1.2.3.13"                              4
  snmpEnableAuthenTraps 1
  saDefaultTimeout    10
  saMaxTimeout  700
# saAllowDuplicateIDs must be set to 1 to allow multiple DPI version 1
#  subagents
  saAllowDuplicateIDs   1
  dpiPathNameForUnixStream "/tmp/dpi_socket"
# Default value of sysServices indicates support for
#  internet, end-to-end, and application layers as
#  defined in RFC 1907.
  sysServices       76
```

The following items explain Example 4-5:

- 1, 2, and 3 You can set these values to meaning names and descriptions for your systems.
- 4 Do not code sysObjectID, it must remain commented.

Determine the type of security supported by the SNMP agent

You must choose to use either community-based security or a combination of community-based and user-based security. There are a number of issues to be considered in making the determination. Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for assistance in making that decision.

- **Community-based security**
  - If you intend to use community-based security (SNMPv1 and SNMPv2c), then set up the PW.SRC and SNMPTRAP.DEST files. The PW.SRC data set and the SNMPD.CONF data set are mutually exclusive. Verify that there is no SNMPD.CONF file because this file can only be used with SNMPv3. If an SNMPD.CONF file is found, the PW.SRC file will not be used.
  - Traps are unsolicited messages that are sent by an SNMP agent to an SNMP network management station. An SNMP trap contains information about a significant network event. To use traps, you must provide SNMPTRAP.DEST information defining a list of managers to which traps are sent. The SNMPTRAP.DEST data set and the SNMPD.CONF data set are mutually exclusive. Verify that there is no SNMPD.CONF file. If an SNMPD.CONF file is found, the SNMPTRAP.DEST file will not be used.
- **Community-based and user-based security**
  - The SNMPD.CONF file defines the SNMP agent security and notification destinations. If the SNMPD.CONF file exists, the agent can support SNMPv1, SNMPv2c, and SNMPv3 requests. If no SNMPD.CONF file exists, the agent will support only SNMPv1 and SNMPv2c requests. A sample SNMPD.CONF file is shipped as /usr/lpp/tcpip/samples/snmpd.conf.

Archived

Note: Do not be confused by the names of the sample file system. The /snmpd.conf is used by the SNMP agent we are now discussing. The /snmpv2.conf is used by the z/OS UNIX **osnmp** command when acting as a manager submitting a request to an agent, and is discussed later:

```
/snmpd.conf is referred to as SNMPD.CONF, and used by the agent
/osnmp.conf is referred to as OSNMP.CONF, and used by the command
/snmpv2.conf is referred to as OSNMP.CONF, and used by the command
```

– The SNMP agent uses the SNMPD.BOOTS configuration file to support SNMPv3 security. This file contains agent information used to authenticate the SNMPv3 requests. The SNMPD.BOOTS keeps the agent identifier and the number of times the agent reboots. If no SNMPD.BOOTS file exists when the agent is started, the agent creates one.

Create the community name file for the SNMP agent

Example 4-6 shows our PWSRC file. Notice that we entered all the community names in upper and lower case to avoid case sensitive issues and *not found* conditions: **jos9m2ap** is the name we are focused on in our scenario.

Example 4-6 SNMP PW.SRC file

```
BROWSE    TCPIPB.TCPPARMS(PWSRC31) - 01.00          Line 00000000 Col 001 080
public  0.0.0.0   0.0.0.0
NSS 10.15.97.0       255.255.255.0
nss 10.15.97.0       255.255.255.0
ral 10.20.0.0         255.255.0.0
RAL 10.20.0.0         255.255.0.0
j0s9m2ap 10.0.0.0       255.0.0.0
JOS9M2AP 10.0.0.0       255.0.0.0
```

Create the trap destination file for the SNMP agent

Example 4-7 shows our trap destination file. For each system, the destinations are the *other* systems in the sysplex that are enabled (configured) to receive traps from this system.

Example 4-7 SNMPTRAP.DEST files for the SNMPD agent on each system

```
BROWSE    TCPIPA.TCPPARMS(TRPDST30) - 01.00         Line 00000000 Col 001 080
# IP ADDRESSES OF THE SYSTEMS WHERE TRAPS CAN BE FORWARDED TO
10.1.1.20 UDP
10.1.1.30 UDP
10.1.1.40 UDP

BROWSE    TCPIPB.TCPPARMS(TRPDST31) - 01.00         Line 00000000 Col 001 080
# IP ADDRESSES OF THE SYSTEMS WHERE TRAPS CAN BE FORWARDED TO
10.1.1.10 UDP
10.1.1.30 UDP
10.1.1.40 UDP
```

Configure the query engine started task (optional)

Update the SNMPQE cataloged procedure by copying the sample in SEZAINST(SNMPPROC) to your system PROCLIB. Specify SNMP parameters, change the

data set names as required to suit your local configuration, and refer to Example 4-8 for a sample of the SNMPQE Proc JCL that we used.

Example 4-8 SNMPQE Proc JCL

```
//SNMPQEB PROC MODULE=SQESERV,PARMS=''
//SNMPQEB EXEC PGM=&MODULE,PARM='&PARMS',
//             REGION=0M,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZADSIL,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.),DISP=SHR
```

Create the SNMPARMS file to be use by the Netview query engine

- Example 4-9 shows NetView SNMPARMS for use with our query engine. Place this member into Netview's DSIPARM dataset.

Example 4-9 SNMPARMS NetView member interface to SNMPQE

```
SNMPQE   SNMPQEB  * Started task name of SNMP Query Engine        1
SNMPQERT 60       * Retry timer (seconds) for IUCV CONNECT
SNMPRCNT 2        * Retry count for sending SNMP requests
SNMPRITO 10       * Retry initial timeout (10ths of a second)
SNMPRETO 2        * Retry backoff exponent (1=linear, 2=exponential)
SNMPMMLL 80       * Line length for Multiline Messages 38/44
```

The following discussion explains Example 4-9:

1 The SNMPQE keyword specifies the actual *started task name* of the SNMP Query Engine. Do not set it to the userID that the started task is running under.

Configure the trap forwarder started task and its files (optional)

The trap forwarder task forwards traps from the SNMP agent to network management applications. It listens for traps on a UDP port (typically 162), and forwards them to all configured managers. If you want to forward trap information to one or more managers, you must configure the trap forwarder. Set up the trap forwarder's started task JCL, its environment variable file, and its configuration file.

> **Note:** If you are going to run both the SNMP Query Engine started task and the trap forwarder started task (both listen on UDP port 162 by default), then trap forwarder will have to listen on a different port from 162 to avoid conflicts. An alternative would be to let them both listen on port 162, but make the trap forwarder to be a bind-specific listener by coding BIND on its port reservation statement and specifying a dynamic vipa address for the forwarder. A dynamic vipa address used for the BIND is specified with the VIPARANGE statement. Client threads sending packets to the forwarder would then specify that bind address to contact the forwarder.

Configure the Trap Forwarder started task JCL

Example 4-10 shows the Trap Forwarder procedure JCL.

Example 4-10 Trap Forwarder proc JCL, TRAPFWDB

```
BROWSE    TCPIPB.TCPPARMS(TRAPFWDB) - 01.01  Line 00000000 Col 001 080
//TRAPFWDB PROC PARMS='-d 0',STDENV=TRPENV&SYSCLONE.                    1
//TRAPFWDB EXEC PGM=EZASNTRA,REGION=0M,TIME=NOLIMIT,
//         PARM=('POSIX(ON) ALL31(ON)',
```

```
//         'ENVAR("_CEE_ENVFILE=DD:STDENV")/-p 1162 &PARMS')
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(&STDENV.)              2
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*
```

The following items explain Example 4-10:

- 1 The &SYSCLONE system variable is used to determine the name of the TRAPENV member that contains settings for environment variables used by the trap forwarder.
- 2 The STDENV DD statement should specify a PDS that has a variable record format (RECFM=V or VB).

Configure the environment variable files for the Trap Forwarder

Example 4-11 shows the environment variables file for the Trap Forwarder.

Example 4-11 STDENV file for the TRAPFWDB task on each system

```
BROWSE    TCPIP.SC30.STDENV(TRPENV30) - 01.02          Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA                  1
RESOLVER_CONFIG=//'TCPIPA.TCPPARMS(DATAA30)'      2
TRAPFWD_CONF=//'TCPIPA.TCPPARMS(TRPCFG30)'        3

BROWSE    TCPIP.SC31.STDENV(TRPENV31) - 01.02          Line 00000000 Col 001 080
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
TRAPFWD_CONF=//'TCPIPB.TCPPARMS(TRPCFG31)'
```

The following items explain Example 4-11:

- 1 _BPXK_SETIBMOPT_TRANSPORT establishes stack affinity to the indicated stack.
- 2 RESOLVER_CONFIG points to the TCPDATA resolver file for the same stack.
- 3 SNMPTRAP_DEST points to the optional trap destination file if needed by the agent

Configure the configuration files for the Trap Forwarder

Example 4-12 shows the forwarder's configuration file. The IP addresses in each file are the addresses of the *other* systems to which traps can be forwarded.

Example 4-12 TRAPFWD.CONF files for the TRAPFWDB task on each system

```
BROWSE    TCPIPA.TCPPARMS(TRPCFG30) - 01.00          Line 00000000 Col 001 080
 10.1.1.20  161
 10.1.1.30  161

BROWSE    TCPIPB.TCPPARMS(TRPCFG31) - 01.01          Line 00000000 Col 001 080
 10.1.1.10  161
 10.1.1.30  161
```

For Trap Forwarder setup and configuration details, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

4.2.3 Activation and verification of the z/OS SNMP agents

If the SNMP agent encounters any errors processing its configuration files, error messages are written to syslogd, not to the console. To activate the SNMP agent, the query engine, and the trap forwarder, enter a start command for each, as shown in Example 4-13.

Example 4-13 Starting the SNMP agent, the query engine, and the trap forwarder

```
S SNMPDB
$HASP373 SNMPDB   STARTED
SNMPDB issues:
   EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE

S SNMPQEB
$HASP100 SNMPQEB  ON STCINRDR
IEF695I START SNMPQEB  WITH JOBNAME SNMPQEB  IS ASSIGNED TO USER TCPIP   , GROUP  TCPGRP
$HASP373 SNMPQEB  STARTED
EZA6275I SNMP Query Engine running and awaiting queries...

S TRAPFWDB
TRAPFWD issues its own startup message, not related to SNMPD startup:
EZZ8409I TRAPFWD: INITIALIZATION COMPLETE
```

After verifying that each started task initializes successfully and issues the expected startup messages, use the NETSTAT CONN command to verify that each one is listening on its respective port, as shown in Example 4-14.

Example 4-14 Netstat CONN display of SNMPD, SNMPQE, and TRAPFWD

```
D TCPIP,TCPIPB,N,CONN
. . .
SNMPDB   0000280E LISTEN
  LOCAL SOCKET:   ::..1029
  FOREIGN SOCKET: ::..0
SNMPDB   0000280D UDP
  LOCAL SOCKET:   ::..161
  FOREIGN SOCKET: *..*
. . .
SNMPQEB  00002815 LISTEN
  LOCAL SOCKET:   0.0.0.0..1030
  FOREIGN SOCKET: 0.0.0.0..0
SNMPQEB  00002813 UDP
  LOCAL SOCKET:   0.0.0.0..6426
  FOREIGN SOCKET: *..*
SNMPQEB  00002814 UDP
  LOCAL SOCKET:   0.0.0.0..162
  FOREIGN SOCKET: *..*
. . .
TRAPFWDB 00002818 UDP
  LOCAL SOCKET:   ::..1162
  FOREIGN SOCKET: *..*
TRAPFWDB 00002819 UDP
  LOCAL SOCKET:   ::..6428
  FOREIGN SOCKET: *..*
. . .
```

Archived

4.3 z/OS SNMP subagents

A subagent extends the set of MIB variables supported by an SNMP agent. Each subagent must be set up and configured independently. The z/OS SNMP subagent and its relationship with the agent and the command client is illustrated in Figure 4-4 on page 132.

The z/OS SNMP subagent topics discussed in this section are:

- "Description of SNMP subagents" on page 142
- "Configuration of SNMP subagents" on page 143
- "Activation and Verification of SNMP subagents" on page 145

4.3.1 Description of SNMP subagents

A subagent extends the set of MIB variables supported by an SNMP agent. Each subagent must be set up and configured independently. The most common subagents associated with the z/OS Communications Server are discussed here.

TCP/IP subagent

The TCP/IP subagent in the z/OS Communications Server is a z/OS UNIX application that runs as a subtask in the TCP/IP address space.

Besides providing support for retrieval of TCP/IP stack management data, the TCP/IP subagent provides SET support, enabling remote configuration of some TCP/IP address space parameters. The TCP/IP subagent is configured and controlled by the SACONFIG statement in the PROFILE.TCPIP data set. If no SACONFIG statement is provided, then the TCP/IP subagent will be started at stack initialization. If you will not be retrieving TCP/IP stack management data, or you do not require the capability to do remote configuration of TCP/IP address space parameters, you can disable the subagent and save system resources.

OMPROUTE subagent

The OMPROUTE subagent implements the Open Shortest Path First (OSPF) MIB variable containing OSPF protocol and state information. The OMPROUTE subagent supports selected MIB objects defined in RFC 1850. The ROUTESA_CONFIG statement is used to enable the subagent support for OMPROUTE.

TN3270 Telnet subagent

The SNMP TN3270 Telnet subagent provides Telnet transaction data for monitored Telnet connections using the SNMP protocol. The TNSACONFIG statement in the TN3270 profile is used to enable the subagent support for TN3270 connections. The TNSACONFIG statement can only be coded within the TELNETGLOBALS block. The TN3270 SNMP MIB module (mvstn3270.mi2) defines performance data for TN3270 connections. The performance data is gathered only if the TN3270E Telnet server has been configured to monitor connections. In order to create and retrieve this data you must do the following:

- Define the MONITORGROUP and MONITORMAP profile statements to the TN3270E Telnet server, mapping the monitor parameters to client ID groups. See Example 4-27 on page 151 and Example 4-28 on page 152.
- Establish TN3270 connections where the client connections match the Client_Identifier parameter specified on one or more of the MONITORMAP profile statements.

Note: The TN3270 MIB objects are available only when monitoring has been defined within the TN3270 profile. If no monitoring has been defined in the profile, then the **`osnmp`**/**`snmp`** command will receive no information to its inquiry.

Network SLAPM2 subagent

The Network SLAPM2 (nslapm2) subagent contains the information that can be used to analyze network performance for respective policy. This subagent runs as its own started task and has SAF security requirements. See the security discussion earlier in this book in "Update RACF to define the SNMP agent started task" on page 134. For details on configuring this subagent, refer to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535.

OSA-Express Direct subagent

The OSA product also provides an SNMP subagent that supports management data for OSA-Express features, called he OSA-Express Direct subagent. It can be used with the z/OS Communications Server SNMP support to retrieve management data.

An SNMP subagent exists on an OSA-Express feature, which is part of a direct path between the z/OS master agent (TCP/IP stacks) and an OSA-Express Management Information Base (MIB). The OSA-Express features support an SNMP agent by providing data for use by an SNMP management application, such as Tivoli NetView. This data is organized into MIB tables defined in the TCP/IP enterprise-specific MIB, as well as standard RFCs. The data is supported by the SNMP TCP/IP subagent. This subagent runs as its own started task and has SAF security requirements. See the security discussion earlier in this book in "Update RACF to define the SNMP agent started task" on page 134.

Additionally, IBM provides a support MIB for the OSA-Express Direct subagent outside of the z/OS Communications Server product and it must be acquired separately. The MIB data supported by the OSA-Express Direct subagent is defined in the OSA enterprise-specific MIB module, IBM-OSA-MIB. This MIB contains the SNMPv2 syntax for the OSA Direct specific MIB objects. The MIB module for the OSA-Express Direct subagent must match your System z server's micro-code level (MCL). The file can be acquired and installed by using one of these two methods:

- Selecting Advanced Functions on the Support Element

  or

- Downloading from:
  https://www-1.ibm.com/servers/resourcelink/lib03010.nsf/0/A310C9113AA36A6485256BBA00697BDA?OpenDocument

  a. After logging in, select **Library**.

  b. Under Library shortcuts on the right side of the screen, select **Open System Adapter (OSA) Library**.

  c. Select **OSA-Express SNMP Direct MIB Module** for a description, or click **TXT** for the module.

  d. Save the MIB file to the location required by your SNMP management application.

> **Note:** If you subscribe to the document "OSA-Express Direct SNMP MIB module" through Resource Link™, then you will receive e-mail notification of document changes.

4.3.2 Configuration of SNMP subagents

Example 4-15 shows how to enable three of the subagents. For complete syntax and additional parameters for each of the statements used, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Enable subagents for TCP/IP stack, OMPROUTE, and TN3270 server,

Example 4-15 Enabling subagents within their respective configuration profiles

```
TCP/IP stack subagent:
   SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161

OMPROUTE subagent:
   ROUTESA_CONFIG ENABLED=YES COMMUNITY="j0s9m2ap" AGENT=161;

TN3270E Telnet server subagent:
   TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
```

Enable subagent for SLA Performance monitor (NSLAPM2)

An example of the NSLAPM2 procedure JCL is shown in Example 4-16. We called our started task SLAPM2B.

Example 4-16 NSLAPM2 Proc JCL, SLAPM2B, subagent setup and its STDENV file

```
BROWSE    SYS1.PROCLIB(SLAPM2B) - 01.02             Line 00000000 Col 001 080
//SLAPM2B PROC STDENV=SLAENV&SYSCLONE.                    1
//SLAPM2B EXEC PGM=NSLAPM2,REGION=0M,TIME=NOLIMIT,
//       PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_CEE_ENVFILE=DD:STDENV")',
//            '/-o -c j02s9m2ap -P 161 -p TCPIPB')          2
//STDENV   DD DSN=TCPIP.SC31.STDENV(&STDENV.),DISP=SHR      3
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)

BROWSE    TCPIP.SC31.STDENV(SLAENV31) - 01.01               1
********************************** Top of Data ***
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
LIBPATH="/usr/lpp/tcpip/lib"
****************************** Bottom of Data *
```

The following items explain Example 4-16:

- 1 The STDENV member contains environment variable settings for NSLAPM2
- 2 The parameters tell NSLAPM2 what agent, community, port, and stack to use
- 3 The STDENV dataset must have RECFM=V or VB.

Enable subagent for OSA-Express Direct subagent

The OSA-Express Direct subagent SNMP support for z/OS is provided by procedure IOBSNMP. This procedure is included as part of the z/OS Communications Server product. However, the MIB is not. You can update the catalogued procedure by copying the sample found in *hlq*.SEZAINST(IOBSNMP) to your system PROCLIB. Change the data set names as required to suit your local configuration. IOBSNMP has four optional parameters, as shown in Example 4-17. We called our started task SNMPOSAB.

Example 4-17 IOBSNMP Proc JCL, SNMPOSAB, setup information about the PARM statement

```
BROWSE    SYS1.PROCLIB(SNMPOSAB) - 01.00            Line 00000000 Col 001 080
//SNMPOSAB PROC PARMS='-d 0'
//IOBSNMP  EXEC PGM=IOBSNMP,TIME=1440,REGION=0M,DYNAMNBR=5,
//         PARM='-c j0s9m2ap -p 161 -s TCPIPB &PARMS.'
//SYSPRINT DD  SYSOUT=*
```

```
//SYSUDUMP DD  SYSOUT=*
```

4.3.3 Activation and Verification of SNMP subagents

We started SLAPM2B using the MVS Start command:

```
S SLAPM2B
```

Typical startup messages for NSLAPM2 are shown in Example 4-18.

Example 4-18 SLAPM2B startup messages

```
. . .
23.57.16 STC08401  IEF695I START SLAPM2B  WITH JOBNAME SLAPM2B  IS ASSIGNED TO U
23.57.16 STC08401  $HASP373 SLAPM2B  STARTED
23.57.16 STC08401  +EZZ8230I NSLAPM2 STARTING ON TCPIPB
23.57.16 STC08401  +EZZ8231I NSLAPM2 CONNECTED TO POLICY AGENT ON TCPIPB
....
main:  EZZ8230I NSLAPM2 STARTING ON TCPIPB
doPAPIConnect:  EZZ8231I NSLAPM2 CONNECTED TO POLICY AGENT ON TCPIPB
```

Some of the verification actions you can perform for the agent and forwarder are discussed in the following sections:

- Review the agent and subagent initialization messages
- Review IOBSNMP initialization and termination messages
- Review SNMPQE initialization and termination messages
- Interface display and traffic monitoring
- TCP/IP stack display and management
- SNMP agent configuration display and management
- Performance statistics displays and monitoring

Review the agent and subagent initialization messages

When the SNMP agent starts it issues an initialization complete message. The subagents also issue their own initialization complete messages. When the subagents successfully connect to the agent each one issues an additional message indicating connection to the agent. The messages indicate whether this is the first time they have connected to the agent while they have been executing or if this is a reconnect. A reconnect occurs if the SNMP agent terminates and then later reinitializes. Samples of the initialization messages are shown in Example 4-19.

Example 4-19 SNMP agent and subagent initialization messages

```
S SNMPDB
$HASP373 SNMPDB   STARTED

SNMPDB issues:
   EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE

OMPB issues:
   EZZ8101I OMPROUTE SUBAGENT INITIALIZATION COMPLETE
   or
   EZZ8108I OMPROUTE SUBAGENT: RECONNECTED TO SNMP AGENT

TN3270B issues:
   EZZ6041I TELNET SNMP SUBAGENT INITIALIZATION COMPLETE
   or
   EZZ6043I TELNET SNMP SUBAGENT RECONNECTED TO SNMP AGENT
```

```
TCPIPB issues:
   EZZ3202I SNMP SUBAGENT: INITIALIZATION COMPLETE
   EZZ3221I SNMP SUBAGENT: SET REQUESTS DISABLED
   or
   EZZ3217I SNMP SUBAGENT: RECONNECTED TO SNMP AGENT

TRAPFWD issues its own startup message, not related to SNMPD startup:
   EZZ8409I TRAPFWD: INITIALIZATION COMPLETE
```

Samples of the termination messages of the agent and subagent are shown in Example 4-20.

Example 4-20 SNMP agent and subagent termination messages

```
P SNMPDB

SNMPDB issues:
   EZZ6204I SIGTERM RECEIVED FOR SNMP DAEMON WHICH IS NOW SHUTTING DOWN
   $HASP395 SNMPDB   ENDED

OMPB issues:
   EZZ8107I OMPROUTE SUBAGENT: CONNECTION TO SNMP AGENT DROPPED

TCPIPB issues:
   EZZ3216I SNMP SUBAGENT: LOST CONNECTION TO SNMP AGENT

TN3270B issues:
   EZZ6042I TELNET SNMP SUBAGENT LOST CONNECTION TO SNMP AGENT

IOBSNMP (SNMPOSAB) issues:
   IOB033E 09/29/2007 16:45:15 SNMP RC -5. Disconnecting from Agent
   IOB002I 09/29/2007 16:45:33 Could not obtain handle from agent. Exiting
```

Review IOBSNMP initialization and termination messages

An example of starting and ending the IOBSNMP started task is shown in Example 4-21. Note that IOBSNMP does not continue to execute if the SNMP agent task is not available or is taken down. The other subagents simply report the loss of contact, and wait until the agent returns. The IOBSNMP task must be started again after the SNMP task is restarted.

Example 4-21 IOBSNMP initialization and termination messages

```
S SNMPOSAB
IOB000I 09/29/2007 16:40:07 Starting OSA SNMP subagent
. . .
IOB028I 09/29/2007 16:40:07 Using stack name TCPIPB
IOB021I 09/29/2007 16:40:07 OSA SNMP subagent initialization complete

P SNMPOSAB
IOB031E 09/29/2007 16:55:24 OSA SNMP subagent has ended
```

Review SNMPQE initialization and termination messages

Both the SNMPQE task and the TRAPFWD task listen on port 162 by default. If you plan to run both, then set up TRAPFWD to listen on a different port. We ran our TRAPFWDB task on port 1162. The startup and shutdown SNMPQE messages are shown in Example 4-22.

Example 4-22 SNMPQE initialization and termination messages

```
S SNMPQEB
$HASP100 SNMPQEB  ON STCINRDR
```

```
IEF695I START SNMPQEB  WITH JOBNAME SNMPQEB  IS ASSIGNED TO USER TCPIP   , GROUP  TCPGRP
$HASP373 SNMPQEB  STARTED
EZA6275I SNMP Query Engine running and awaiting queries...

P SNMPQEB
$HASP395 SNMPQEB  ENDED
```

Interface display and traffic monitoring

Refer to 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 149 for information about how to use the **snmp** command. By monitoring the amount of data transmitted over a router's interfaces, an administrator can monitor how many bytes of data have been transmitted between IP subnetworks in a particular period of time. You can also monitor the traffic size over an interface of an IP host that is running as an application server. The following are examples of MIB objects that can be used for this monitoring:

- ifInOctets gives the total number of octets received on the interface, including framing characters.
- ifOutOctets gives the total number of octets transmitted out of the interface, including framing characters.
- sysUpTime provides a count in one hundredths of a second of how long the SNMP agent has been running. That is how long the device has been up and running in most cases. This value is useful to determine the time differences from the previous collection and whether previous counter information is valid. Therefore, this value should be retrieved with each collection.

TCP/IP stack display and management

Refer to 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 149 for information about how to use the **snmp** command. By changing particular MIB values, a TCP/IP stack configuration can be altered without recycling. Some of the operations supported are:

- Change the IP forwarding attributes.
- Change the logic of the equal-cost multipath activation.
- Drop an existing TCP connection established to a remote IP host.
- Alter the buffer size allocated for each TCP connection or UDP association.

SNMP agent configuration display and management

Refer to 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 149 for information about how to use the **snmp** command. The SNMPv3 framework allows you to change the SNMP agent's configuration dynamically. Some example operations supported are:

- Change the security keys for SNMP managers.
- Add new users or SNMP managers.
- Change the group definition.

Performance statistics displays and monitoring

Refer to 4.4.3, “Using the osnmp/snmp z/OS UNIX command” on page 149 for information about how to use the **snmp** command. The SLA subagent maintains the SLA Performance Monitor MIB-2 (SLAPM2) that gives various performance information for each policy installed in a TCP/IP stack. Some possible monitoring options are:

- Monitor the throughput.
- Monitor the transmission delay.
- Monitor the amount of data that meet a policy specification.

4.4 z/OS SNMP client command

The z/OS Netview SNMP and snmp commands act as a manager requesting information from an agent. The two SNMP client applications provided with z/OS Communications Server are:

- SNMP command from the NetView environment
- **snmp** command in the z/OS shell

4.4.1 Description of the SNMP client commands

The SNMP command in the NetView environment requires the use of the NetView product. It supports SNMP version 1. The **snmp**/**osnmp** command in the z/OS shell supports SNMP versions 1, 2, and 3. Depending on your requirements, you might decide to configure either or both of these clients, or to use an SNMP client on another platform. A brief description of the required implementation steps is given here. For complete details, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

4.4.2 Configuration tasks for the SNMP client commands

The configuration tasks for both the Netview SNMP command and the z/OS UNIX **snmp** command are discussed in this section.

Implementation of the Netview SNMP command

The NetView environment requires the set up of a number of items in the NetView address space, such as the SNMPIUCV task, the SNMP command processor, and the SNMP message definition file.

Perform the following tasks to prepare the Netview SNMP command:

1. Set up the SNMP Query Engine started task
2. Set up the SNMPARMS control member for NetView
3. Set up the hlq.MIBDESC.DATA data set

Set up the SNMP Query Engine started task

Update the SNMPQE cataloged procedure by copying the sample in SEZAINST(SNMPPROC) to your system PROCLIB. Specify SNMP parameters and change the data set names as required to suit your local configuration. Example 4-8 on page 139 shows a sample of the SNMPQE started task JCL.

Set up the SNMPARMS control member for NetView

SNMPIUCV reads the SNMPARMS member in the SEZADSIP data set at startup. This data set contains the initialization parameters for SNMP. The data set containing SNMPARMS should be added to the DSIPARM DD statement in the NetView startup procedure. Example 4-9 on page 139 shows NetView SNMPARMS for use with our query engine.

Set up the hlq.MIBDESC.DATA data set

The SNMP Query Engine (SQESERV) needs access to the *hlq*.MIBDESC.DATA data set for the MIB variable descriptions. You can find a sample of this data set in SEZAINST(MIBDESC) to copy into your dataset. When creating your *hlq*.MIBDESC.DATA dataset, make sure that *hlq* matches what you have coded as the DATASETPREFIX in the TCPDATA resolver file.

Implementation of the osnmp z/OS UNIX command

The **osnmp** command is used to send SNMP requests to SNMP agents on local or remote hosts. The requests can be SNMPv1, SNMPv2, or SNMPv3. For SNMPv2 and SNMPv3

requests, the OSNMP.CONF configuration file is required. The *winSNMPname* specified on an OSNMP.CONF statement can be used as the value of the *-h* parameter on the **osnmp** command.

Perform the following tasks to prepare the **snmp**/**osnmp** command:

1. Set up snmp configuration information.
2. Set up user MIB object information.

Set up snmp configuration information

The OSNMP.CONF file is used to define target agents and, for SNMPv3, the security parameters to be used in sending requests to them. The contents of the file, either /etc/osnmp.conf or /etc/snmpv2.conf, regardless of location, are the same. Only the first file found is used. A sample of this file is installed as /usr/lpp/tcpip/samples/snmpv2.conf. This sample should be copied and modified for your installation. The file we used is shown in Example 4-23.

Example 4-23 /etc/snmpv2.conf file used by the osnmp/snmp z/OS UNIX command

```
#-------------------------------------------------------------
# Community-based security  (SNMPv1 and SNMPv2c)
#-------------------------------------------------------------
v1       127.0.0.1   snmpv1
v2c      127.0.0.1   snmpv2c
v2c_ipv6 ::1         snmpv2c
mvs1     10.67.113.79 snmpv2c
sc30b    10.1.1.10 snmpv2c
sc31b    10.1.1.20 snmpv2c
sc32b    10.1.1.30 snmpv2c
# mvs2   mvs2c       snmpv2c    nosvipa
# mvs3   mvs3:1061   snmpv2c
mvs4     12ab::2     snmpv2c
```

Set up user MIB object information

If you want to use the textual names for MIB objects that are not defined in the compiled MIB, then you can define them to the **snmp** command using the MIBS.DATA file. A sample of the MIBS.DATA file is installed as /usr/lpp/tcpip/samples/mibs.data. Copy this sample to /etc/mibs.data, and modify it for your installation.

4.4.3 Using the osnmp/snmp z/OS UNIX command

Refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781, for information about how to use the **osnmp**/**snmp** command. Use the getbulk option of the **snmp** command to retrieve multiple objects with one command. A few examples of using the **snmp** command follow.

> **Tip:** If you are interested in a single MIB object only, then you should use the **get** option with the fully qualified MIB object, instead of the **getbulk** option as shown in our examples.

Example 4-24 shows system information. Notice that the Agent and active Subagents are represented in the list by the *sysORDescr.x* entries

Example 4-24 snmp system information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v getbulk 1.3.6.1.2.1.1
myDescr.0 = SNMPv3 agent version 1.0 with DPI version 2.0
myObjectid.0 = 1.3.6.1.4.1.2.3.13
```

```
myUptime.0 = 34300
myContact.0 = I&O Mainframe Network Support
myName.0 = SC30 - z/OS Communications Server
myLocation.0 = Datacenter Mainframe Operations
myServices.0 = 76
sysORLastChange.0 = 6500
sysORID.1 = 1.3.6.1.4.1.2.11.7.1
sysORID.2 = 1.3.6.1.4.1.2.11.7.2
sysORID.3 = 1.3.6.1.4.1.2.11.7.6
sysORID.4 = 1.3.6.1.4.1.2.11.7.3
sysORID.5 = 1.3.6.1.4.1.2.11.26.1
sysORDescr.1 = z/OS SNMP Agent
sysORDescr.2 = z/OS TCP/IP SNMP Subagent
sysORDescr.3 = z/OS TN3270 SNMP Subagent
sysORDescr.4 = z/OS OSPF SNMP Subagent
sysORDescr.5 = OSA subagent

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v getbulk 1.3.6.1.2.1.1
myDescr.0 = SNMPv3 agent version 1.0 with DPI version 2.0
myObjectid.0 = 1.3.6.1.4.1.2.3.13
myUptime.0 = 15400
myContact.0 = I&O Mainframe Network Support
myName.0 = SC31 - z/OS Communications Server
myLocation.0 = Datacenter Mainframe Operations
myServices.0 = 76
sysORLastChange.0 = 700
sysORID.1 = 1.3.6.1.4.1.2.11.7.1
sysORID.2 = 1.3.6.1.4.1.2.11.26.1
sysORID.3 = 1.3.6.1.4.1.2.11.7.6
sysORID.4 = 1.3.6.1.4.1.2.11.7.2
sysORID.5 = 1.3.6.1.4.1.2.11.7.3
sysORDescr.1 = z/OS SNMP Agent
sysORDescr.2 = OSA subagent
sysORDescr.3 = z/OS TN3270 SNMP Subagent
sysORDescr.4 = z/OS TCP/IP SNMP Subagent
sysORDescr.5 = z/OS OSPF SNMP Subagent
CS07 @ SC30:/u/cs07>
```

We established a TN3270 client connection from the SC31 TSO session to the SC30 TN3270E Telnet server. From SC30's perspective the local socket would be 10.1.1.10 and the Foreign socket would be 10.1.1.20, as seen in Example 4-25.

Example 4-25 snmp stack information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v getbulk ibmMvsTcpipProcname
ibmMvsTcpipProcname.0 = TCPIPB
ibmMvsTcpipAsid.0 = 111

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v getbulk ibmMvsTcpipProcname
ibmMvsTcpipProcname.0 = TCPIPB
ibmMvsTcpipAsid.0 = 132

CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 30 getbulk tcpConnectionEntry
tcpConnectionState.1.4.10.1.1.10.23.1.4.10.1.1.20.1029 = 5
tcpConnectionState.1.4.10.1.1.10.1056.1.4.10.1.1.10.1055 = 8
tcpConnectionState.1.4.10.1.1.10.1062.1.4.10.1.1.10.1061 = 8
tcpConnectionState.1.4.10.1.8.11.992.1.4.10.1.6.21.1035 = 5
tcpConnectionState.1.4.127.0.0.1.1024.1.4.127.0.0.1.1025 = 5
tcpConnectionState.1.4.127.0.0.1.1025.1.4.127.0.0.1.1024 = 5
tcpConnectionState.1.4.127.0.0.1.1081.1.4.127.0.0.1.1082 = 5
```

```
tcpConnectionState.1.4.127.0.0.1.1082.1.4.127.0.0.1.1081 = 5

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v -m 30 getbulk tcpConnectionEntry
tcpConnectionState.1.4.10.1.1.20.1029.1.4.10.1.1.10.23 = 5
tcpConnectionState.1.4.127.0.0.1.1024.1.4.127.0.0.1.1025 = 5
tcpConnectionState.1.4.127.0.0.1.1025.1.4.127.0.0.1.1024 = 5
tcpConnectionState.1.4.127.0.0.1.1026.1.4.127.0.0.1.1027 = 5
tcpConnectionState.1.4.127.0.0.1.1027.1.4.127.0.0.1.1026 = 5
```

Example 4-26 on page 151 shows omproute/ospf information.

Example 4-26 snmp Omproute/OSPF information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 10 getbulk ospf
ospfRouterId.0 = 10.1.1.10
ospfAdminStat.0 = 1
ospfVersionNumber.0 = 2

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v -m 10 getbulk ospf
ospfRouterId.0 = 10.1.1.20
ospfAdminStat.0 = 1
ospfVersionNumber.0 = 2

CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 10 getbulk ospfIfTable
ospfIfIpAddress.10.1.1.10.0 = 10.1.1.10
ospfIfIpAddress.10.1.2.11.0 = 10.1.2.11
ospfIfIpAddress.10.1.2.12.0 = 10.1.2.12
ospfIfIpAddress.10.1.3.11.0 = 10.1.3.11
ospfIfIpAddress.10.1.3.12.0 = 10.1.3.12
ospfIfIpAddress.10.1.4.11.0 = 10.1.4.11
ospfIfIpAddress.10.1.5.11.0 = 10.1.5.11
ospfIfIpAddress.10.1.6.11.0 = 10.1.6.11
ospfIfIpAddress.10.1.8.11.0 = 10.1.8.11
ospfIfIpAddress.10.1.8.12.0 = 10.1.8.12

CS07 @ SC30:/u/cs07>snmp -h sc31b -c j0s9m2ap -v -m 10 getbulk ospfIfTable
ospfIfIpAddress.10.1.1.20.0 = 10.1.1.20
ospfIfIpAddress.10.1.2.21.0 = 10.1.2.21
ospfIfIpAddress.10.1.2.22.0 = 10.1.2.22
ospfIfIpAddress.10.1.3.21.0 = 10.1.3.21
ospfIfIpAddress.10.1.3.22.0 = 10.1.3.22
ospfIfIpAddress.10.1.4.21.0 = 10.1.4.21
ospfIfIpAddress.10.1.5.21.0 = 10.1.5.21
ospfIfIpAddress.10.1.6.21.0 = 10.1.6.21
ospfIfIpAddress.10.1.8.21.0 = 10.1.8.21
ospfIfIpAddress.10.1.8.22.0 = 10.1.8.22
```

Example 4-27 on page 151 shows TN3270 information.

Example 4-27 snmp TN3270E Telnet server information

```
CS07 @ SC30:/u/cs07>snmp -h sc30b -c j0s9m2ap -v -m 40 getbulk ibmMvsTN3270ConnTable
ibmMvsTN3270ConnStartTime.1.4.10.1.1.10.23.1.4.10.1.1.20.1031=2007-09-29,0:0:58.8
ibmMvsTN3270ConnAppl.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = SC30TS02
ibmMvsTN3270ConnLuName.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = SC30BB01
ibmMvsTN3270ConnLogMode.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = SNX32704
ibmMvsTN3270ConnProto.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = '02'h
ibmMvsTN3270ConnRtGroupIndex.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 1
ibmMvsTN3270ConnRtIpMethod.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtAvgRt.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
```

```
ibmMvsTN3270ConnRtAvgIpRt.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtAvgCountTrans.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 8
ibmMvsTN3270ConnRtIntTimeStamp.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 2007-09
-29,0:6:58.8
ibmMvsTN3270ConnRtTotalRts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtTotalIpRts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtCountTrans.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 12
ibmMvsTN3270ConnRtCountIP.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtElapsRndTrpSq.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtElapsIpRtSq.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtElapsSnaRtSq.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket1Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 12
ibmMvsTN3270ConnRtBucket2Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket3Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket4Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270ConnRtBucket5Rts.1.4.10.1.1.10.23.1.4.10.1.1.20.1031 = 0
ibmMvsTN3270MonGroupName.1 = SNAONLY
ibmMvsTN3270MonGroupName.2 = SNAANDIP
ibmMvsTN3270MonGroupType.1 = '30'h
ibmMvsTN3270MonGroupType.2 = 'f0'h
ibmMvsTN3270MonGroupSampPeriod.1 = 120
ibmMvsTN3270MonGroupSampPeriod.2 = 120
ibmMvsTN3270MonGroupSampMult.1 = 5
ibmMvsTN3270MonGroupSampMult.2 = 5
ibmMvsTN3270MonGroupBucketBndry1.1 = 100
ibmMvsTN3270MonGroupBucketBndry1.2 = 100
ibmMvsTN3270MonGroupBucketBndry2.1 = 200
ibmMvsTN3270MonGroupBucketBndry2.2 = 200
ibmMvsTN3270MonGroupBucketBndry3.1 = 300
ibmMvsTN3270MonGroupBucketBndry3.2 = 300
ibmMvsTN3270MonGroupBucketBndry4.1 = 400
ibmMvsTN3270MonGroupBucketBndry4.2 = 400
ibmProd.188.1.1.1.1.12 = '0002'h
CS07 @ SC30:/u/cs07>
```

The TN3270E Telnet server profile must have MONITORGROUP and MONITORMAP statements defining the performance statistics, which should be collected for the mapped connections.

Note: The TN3270 MIB objects are available only when monitoring has been defined within the TN3270 profile. If no monitoring has been defined in the profile, then the **osnmp**/**snmp** command will receive no information to its inquiry.

The statements in our TN3270 profile, TELNB30B, are shown in Example 4-28.

Example 4-28 TN3270E Telnet server profile statements defining the MONITORGROUP for port 23

```
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS
MONITORGROUP SNAONLY
    AVERAGE
    BUCKETS
    NODYNAMICDR
    NOINCLUDEIP
    AVGSAMPPERIOD 120
    AVGSAMPMULTIPLIER 5
```

```
    BOUNDARY1 100
    BOUNDARY2 200
    BOUNDARY3 300
    BOUNDARY4 400
  ENDMONITORGROUP

  DESTIPGROUP ALLUSERS    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

  MONITORMAP SNAONLY    DESTIPGRP,ALLUSERS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL SC30N*          ; Netview
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
```

The profile statements defining the MONITORGROUP for port 992 are shown in Example 4-29.

Example 4-29 TN3270E Telnet server profile statements defining the MONITORGROUP for port 992

```
MONITORGROUP SNAANDIP
  AVERAGE
  BUCKETS
  DYNAMICDR
  INCLUDEIP
  AVGSAMPPERIOD 120
  AVGSAMPMULTIPLIER 5
  BOUNDARY1 100
  BOUNDARY2 200
  BOUNDARY3 300
  BOUNDARY4 400
ENDMONITORGROUP

DESTIPGROUP GENERALUSER 10.1.8.11   ENDDESTIPGROUP
DESTIPGROUP ADMIN       10.1.8.12   ENDDESTIPGROUP
DESTIPGROUP PAYROLL     10.1.8.23   ENDDESTIPGROUP
DESTIPGROUP SHIPPING    10.1.1.10   ENDDESTIPGROUP
DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.0   ENDDESTIPGROUP

MONITORMAP SNAANDIP   DESTIPGRP,GENERALUSER
PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER
```

Additional SNMP MIB objects

In previous versions, TCP/IP only used SNMP counters for the number of the accepted TCP connections in the *ibmTcpipMvsTcplistenerTable* MIB table. There was one set of counters per server represented in the table defined in the IBM MVS TCP/IP enterprise-specific MIB modules shipped with z/OS Communication Server and supported by the TCP/IP subagent.

To determine the total count of accepted connections for a TCP/IP stack management applications had to retrieve the accepted connection counter for each server and sum these counts. Accepted connections counters for servers which had terminated were no longer available.

Archived

The z/OS Communication Server provides two specific counters in the IBM MVS TCP/IP enterprise-specific MIB module:

- **ibmMvsTcpAcceptCount** - 32-bit counter
- **ibmMvsTcpHCAcceptCount** - 64-bit counter

Both counters provide the total number of connections accepted by all listeners. They differ only on the size.

Example 4-30 shows samples that we obtained from **snmp** commands in UNIX System Services.

Example 4-30 Examples of TCP layer accepted connections counters.

```
CS07 @ SC30:/u/cs07>snmp -h 10.1.1.20 -c j0s9m2ap -v getbulk ibmMvsTcpHCAcceptCo
unt
ibmMvsUdpLastAct.0.0.0.0.161 = 0
CS07 @ SC30:/u/cs07>snmp -h 10.1.1.20 -c j0s9m2ap -v getbulk ibmMvsTcpAcceptCoun
t
ibmMvsTcpAcceptCount.0 = 1
CS07 @ SC30:/u/cs07>cd \
```

Another object available in the IBM MVSTCP/IP enterprise-specific MIB module is the following:

ibmMvsCpcNd -The Central Processing Complex node descriptor value for the central processing complex on which the TCP/IP subagent is active. This value can be used to uniquely identify this CPC. If the node descriptor cannot be obtained, then this MIB object will be set to a zero-length string.

In Example 4-31 we see that the object *ibmMvsCpcNd* can be obtained in two different ways. One is through the snmp command and the other through the MVS console command D M=CPU.

Example 4-31 The ibmMvsCpcNd obtained in two different ways.

```
CS07 @ SC30:/u/cs07>snmp -h 10.1.1.20 -c j0s9m2ap -v getbulk ibmMvsCpcNd
ibmMvsCpcNd.0 = 'f0f0f2f0f9f4e2f1f8c9c2d4f0f2f0f0f0f0f0f0f0f2f9f9f1c5fff0'h
CS07 @ SC30:/u/cs07>
. . . . . .
D M=CPU
 IEE174I 15.01.12 DISPLAY M 137
 PROCESSOR STATUS
 ID  CPU                 SERIAL
 00  +                    23991E2094
 01  +                    23991E2094
 02  +                    23991E2094
 03  +                    23991E2094

 CPC ND = 002094.S18.IBM.02.00000002991E
 CPC SI = 2094.712.IBM.02.000000000002991E
 CPC ID = 00
 CPC NAME = SCZP101
 LP NAME = A23        LP ID = 23
 CSS ID  = 2
 MIF ID  = 3

 + ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
 N NOT AVAILABLE
```

```
CPC ND  CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI  SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID  CENTRAL PROCESSING COMPLEX IDENTIFIER
CPC NAME CENTRAL PROCESSING COMPLEX NAME
LP NAME  LOGICAL PARTITION NAME
LP ID    LOGICAL PARTITION IDENTIFIER
CSS ID   CHANNEL SUBSYSTEM IDENTIFIER
```

Removal of DPI API 4KB buffer restriction

Previously, there were restrictions when using Distributed Protocol Interface (DPI) API for any SNMP subagent that connected to the z/OS SNMP agent to provide management data. The agent used DPI API to send SNMP requests to subagents and the subagents used the DPI API to send the agent responses to SNMP requests and traps. The DPI API had a hard-coded limit of 4 KB buffer-size. Because of the buffer-size, the management applications using SNMP GETNEXT, GETBULK, or BULKWALK often receive a “too big” error on requests and could not manage the information for the clients.

To allow larger amounts of data to be exchanged between SNMP agent and subagent, DPI API now supports up to 65535 byte buffers and the agents support up to 65535 bytes for responses.

Important: For subagents which are not provided by z/OS Communication Server, the subagent load modules must be linked again to include the latest DPI API functions.

As a result, the applications can retrieve more data per SNMP request, especially when using the SNMP GETBULK and BULKWALK options.

Options for message EZZ6317I

SNMP message EZZ6317I was created to warn that support for configuration of the SNMP sysObjectId value would be discontinued in a future release, however the message was only written to the syslog and it was not always noticed. The sysObjectId MIB object should not be set to other than its default value because it identifies the z/OS Communication Server agent to network management applications.

An option exists to enable the message EZZ6317I to be written to the console and syslog daemon, making the it more visible to those customers currently configuring this MIB object.

Sample of OSNMPD.DATA information is shipped with the z/OS Communication Server product and installed as file /usr/lpp/tcpip/samples/osnmpd.data. OSNMPD.DATA information can be used to configure values of some of the MIB objects supported by the SNMP agent.

Example 4-32 illustrates the sample file where the object sysObjectId was commented out and, in this way, you do not have the message written to the console nor to syslogd.

Example 4-32 OSNMPD.DATA without sysObjectId.

```
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5694-A01
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "Unknown"
  sysLocation "Unknown"
  sysName "z/OS Communications Server"
# Default value of sysObjectID is equivalent to ibmTcpIpMvs
#  in the ibmAgents subtree; this is the sysObjectID representing
```

```
#  IBM z/OS Communications Server
# Changing this value is not recommended, as it is intended to allow
#  network management applications to identify this agent as the
#  z/OS Communication Server SNMP agent.  The ability to change it
#  will be disabled in a subsequent release.
# sysObjectID     "1.3.6.1.4.1.2.3.13"
  snmpEnableAuthenTraps 1
  saDefaultTimeout    6
  saMaxTimeout  700
# saAllowDuplicateIDs must be set to 1 to allow multiple DPI version 1
#  subagents
  saAllowDuplicateIDs   1
  dpiPathNameForUnixStream "/tmp/dpi_socket"
# Default value of sysServices indicates support for
#  internet, end-to-end, and application layers as
#  defined in RFC 1907.
  sysServices       76
```

Example 4-33 is before changing the OSNMPD.DATA file. We did not receive the EZZ6317I message.

Example 4-33 Before changing the OSNMPD.DATA file

```
S SNMPD
$HASP100 SNMPD    ON STCINRDR
IEF695I START SNMPD    WITH JOBNAME SNMPD    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 SNMPD    STARTED
EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE
EZZ3217I SNMP SUBAGENT: RECONNECTED TO SNMP AGENT
```

Example 4-34 shows the startup messages after changing the OSNMPD.DATA file, which enabled the sysObjectId.

Example 4-34 After changing the OSNMPD.DATA file

```
S SNMPD
$HASP100 SNMPD    ON STCINRDR
IEF695I START SNMPD    WITH JOBNAME SNMPD    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 SNMPD    STARTED
EZZ6317I CONFIGURATION OF SYSOBJECTID ACCEPTED BUT WILL NOT BE
ALLOWED IN FUTURE RELEASES
EZZ6225I SNMP AGENT: INITIALIZATION COMPLETE
```

4.5 Problem determination for the SNMP facilities

The z/OS UNIX **osnmp** command provides the SNMP manager function from the z/OS shell in order to query SNMP agents for network management information. Use the **osnmp** command to issue SNMP requests to agents and to process SNMP responses returned by agents. The z/OS UNIX **osnmp** command supports SNMPv1, SNMPv2c, and SNMPv3 requests.

snmp is a synonym for the **osnmp** command in the z/OS UNIX shell. **snmp** command syntax is the same as that for the **osnmp** command.

For a comprehensive discussion on the use of the **osnmp** command and its syntax, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Some MIB values are very useful for problem determination in IP networks to solve problems such as performance problems or lack of connectivity.

Examples of MIB objects that can be used to determine problems in the data-link interface (if) layer are:

- ifInErrors, ifOutErrors: give the number of inbound/outbound frames that were discarded due to errors.
- ifInDiscards, ifOutDiscards: give the number of inbound/outbound frames that were discarded due to resource limitations.

Examples of MIB objects that can be used to determine problems in the IP layer are:

- ipInHdrErrors: gives the number of IP packets that were discarded because of errors in their IP headers.
- ipInUnknownProtos: indicates the number of IP packets that were received successfully but discarded because of either unknown or unsupported protocol.
- ipInDiscards, ipOutDiscards: give the number of inbound/outbound IP packets that were discarded due to resource limitations.
- ipReasmFails: provides the number of failures detected by the IP re-assembly algorithm (for whatever reason: timed out, errors, etc).

The following MIB objects are useful to determine problems related to the TCP or UDP protocol:

- tcpRetransSegs: gives the number of TCP segments retransmitted.
- udpInErrors: gives the number of UDP datagrams discarded for reasons other than the lack of an application at the destination port.

4.6 Additional information sources for SNMP

Detailed information for the SNMP protocols can be found in the following documents:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *OSA-Express Implementation Guide*, SG24-5948
- *TCP/IP Tutorial and Technical Overview*, GG24-3376

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

Archived

5

IP printing

IP printing supports the sending of print output from a client device on an IP network to a printer or a print server gateway on that network. This chapter focuses on the IP printing functions that are available in the z/OS Communications Server.

The protocols defined in Request for Comment (RFC) 1179 and its amendments form the basis for printing in a TCP/IP network. The line printer daemon (LPD) is the remote print server and the line printer requester (LPR) is the client defined by this protocol. If the destination printer is a network printer, one defined to the LPD with an IP address or a DNS name, then the LPD uses the TCP protocol to forward the file to the final IP destination.

This chapter discusses the following IP printing topics.

Section	Topic
5.1, “Conceptual overview of IP printing” on page 160	The basic concepts of IP printing.
5.2, “LPR/LPD” on page 162	Commonly implemented LPR/LPD scenarios, their dependencies, advantages, and configuration examples.
5.3, “Infoprint Server” on page 167	An overview of Infoprint Server and its components. Sample setup ISPF panels are listed (implementation or verification procedures are not shown because they are beyond the scope of this publication).
5.4, “Problem determination for LPR/LPD” on page 176	Problem determination procedures for the LPR/LPD scenario.
5.5, “Additional information sources for IP printing” on page 183	References to additional documentation about IP Printing.

© Copyright IBM Corp. 2008. All rights reserved.

5.1 Conceptual overview of IP printing

As illustrated in Figure 5-1, the line printer requester (LPR) and the line printer daemon (LPD) are standard applications provided with the z/OS Communications Server that support IP Printing Services. The Line Print Requester (client) and the Line Print Daemon (LPD) use the Pascal application programming interface to send output data directly to the TCP/IP protocol layers, bypassing the socket interfaces of the Logical and Physical file systems of z/OS UNIX.

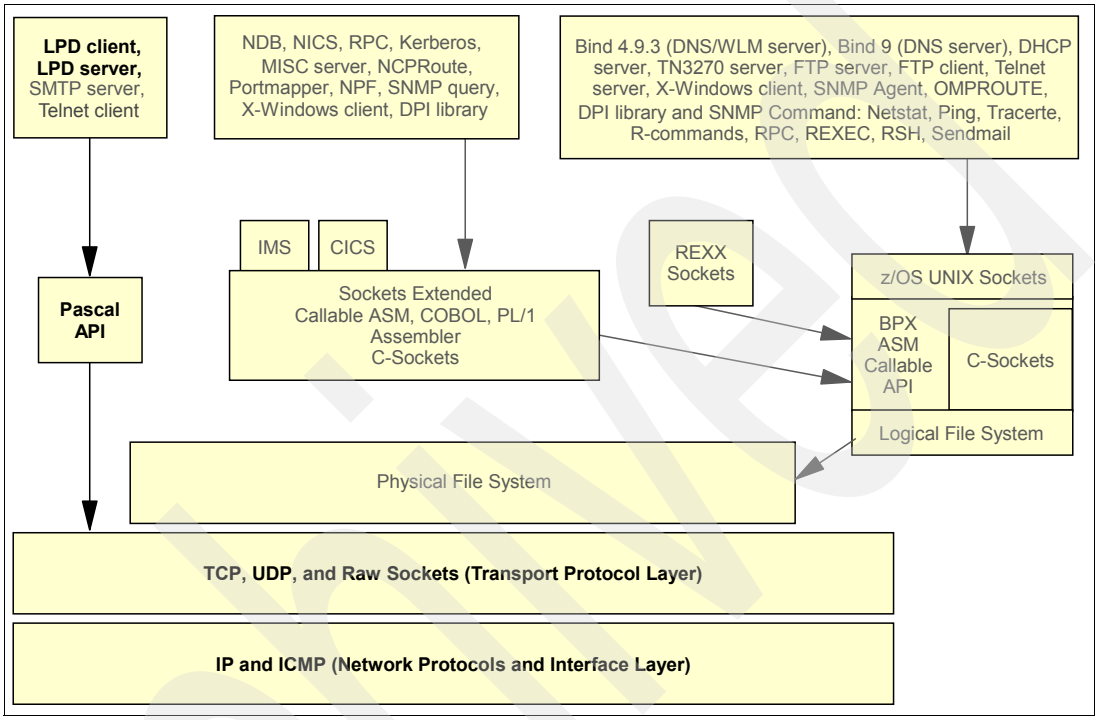


Figure 5-1 z/OS IP Printing Services

Many printers can support LPD themselves, allowing network clients to print directly to them. Also, with advanced implementations, print services can be provided by a dedicated print management solution that provides robust management and administration capabilities for many printers.

This section discusses the following concepts:

- 5.1.1, “What is IP printing?” on page 160
- 5.1.2, “How does IP Printing work?” on page 161
- 5.1.3, “How can IP Printing be applied?” on page 161

5.1.1 What is IP printing?

The basic capabilities of LPR are very limited. LPR is a good testing tool for verifying communications to a print server and for verifying print delivery to a printer controlled by that server. However, if advanced formatting features or print file management is required, additional products are needed. Printing products such as the IBM Infoprint Server extend the options for the management of printing in your organization. The Infoprint Server is the IBM strategic z/OS IP print management product that utilizes IP protocols to deliver centralized enterprise-wide print management. By combining the flexible Infoprint Server interfaces with the robust spool management capabilities of the Job Entry Subsystem (JES) you can achieve state-of-the-art print operations for most IP printing needs.

No future functional enhancements are planned for the IBM Network Print Facility (NPF) for z/OS, the predecessor of Infoprint Server.

5.1.2 How does IP Printing work?

The diagram in Figure 5-2 shows the relationship of LPR to LPD.

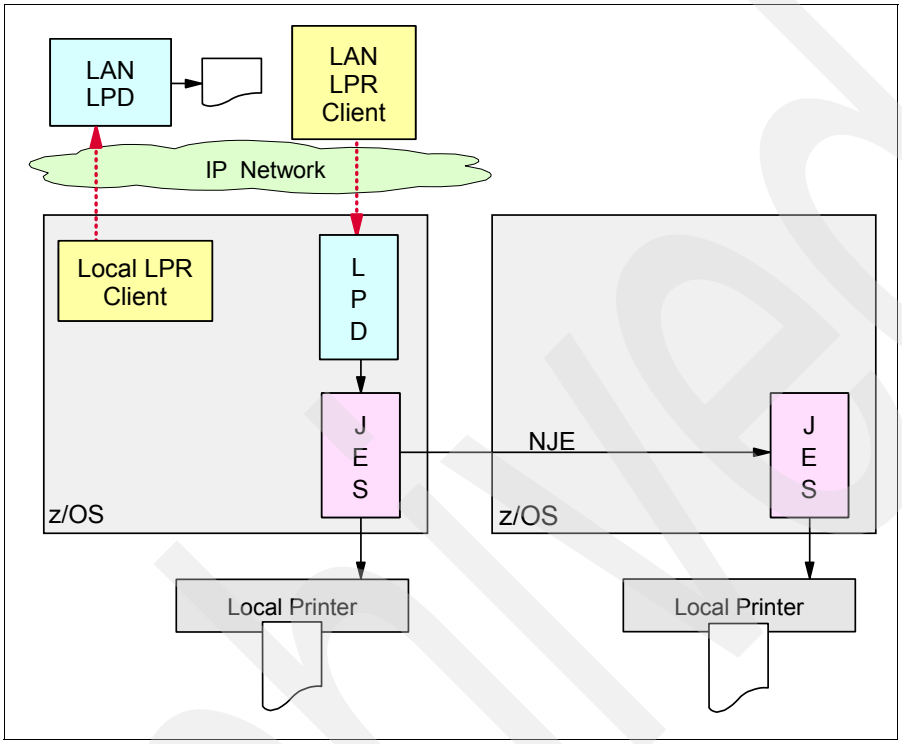


Figure 5-2 LPR/LPD relationships

In a simple LPR/LPD environment with no additional print management products, a client can issue the LPR command to print a file. By specifying a few parameters on the LPR command, the client can direct the output to a printer defined on a print server that is acting as the LPD server. Optional LPR parameters can be used to manipulate the printed file format and to instruct the LPD server to perform additional processing once the file arrives at the server. Two files are transmitted from the LPR client to the LPD server for each file to be printed:

- A control file contains structured parameter settings such as number of copies, special forms, special fonts, target printer, queue class, and many others.
- The actual data file to be printed.

The LPD server is responsible for managing the print queues for printers it has defined. It is also responsible for the integrity of the received files and for successfully printing them. When it is accepting print for a mainframe-attached printer (local system printer or remote printer), the z/OS LPD server uses the JES spool as its repository and takes advantage of spool integrity management provided by JES.

5.1.3 How can IP Printing be applied?

IP printing can be accomplished by using a simple LPR/LPD implementation. More advanced printing packages can be used to provide advanced capabilities, such as documenting and

managing printer inventory, providing data stream transforms, and providing a Web-based application for help desk operators to access print queue status and printer information.

With the advent of TCP/IP networking and newer information technologies, printing requirements are changing. For example:

- Applications and workstation users need the flexibility to print to any printer (network printers and host printers).

  Businesses that print statements (such as banking statements, invoices, and bills of materials) need to print both on network-attached printers and on higher-volume, host-attached printers.
- Users need easy-to-use software.

  Users want graphical interfaces (GUIs) to handle the complex tasks of printing and managing printers.
- Companies require more print server capacity.

  Companies with a combination of stand-alone and host-connected printers need more print server capacity to meet their distributed printing needs.

These requirements introduce new issues:

- How to handle the wide range of printers and formatting options available in an environment and enable users of traditional terminals and distributed workstations share those printers
- How to support print from applications that are consolidated into a z/OS environment without re-engineering their printing functions
- How to reduce costs by consolidating print servers

Possible solutions for these issues can be achieved by using the simple LPR/LPD functions, or by implementing the more sophisticated Infoprint Server facilities.

When a simple LPD server is required for inbound print

If your printing requirements are to provide an LPD server on the z/OS platform with little or no use of the outbound LPR function, then a simple implementation of the LPD server will achieve a quick solution. This supports inbound print traffic that is to be printed by the mainframe. However, it will be limited in functionality.

When manageability of centralized printing is required

If outbound print traffic is a requirement, then the management of that printing cannot be provided with the simple LPR function. A more robust solution will be necessary to provide queue management and reporting. In addition, you may be faced with a requirement to reformat print files before they are delivered to printers requiring special data streams. A printing solution like the IBM Infoprint Server should be considered.

5.2 LPR/LPD

A simple Line Print Daemon (LPD) server can be implemented on the mainframe to enable the z/OS platform as a print server. This section discusses the following topics:

- Description of LPR/LPD
- Configuration tasks for LPR/LPD
- Activation and verification of LPR/LPD

5.2.1 Description of LPR/LPD

You can use LPD with the TSO LPR command or batch LPR jobs to support TCP/IP print. This type of setup is limited in functionality, but involves minimal effort and planning.

A TSO session user can enter the client LPR command manually for each file to be printed. A batch job can also be set up to execute the LPR command in a background batch TSO environment. The job can be scheduled and automated through a job scheduling system, but there is no centralized management of the print files.

Dependencies of LPR/LPD

The z/OS LPD server uses JES to manage print files. The LPD server can receive the print files being forwarded to it by the various LPR clients in the network. If the destination printer is a z/OS controlled printer, the print file is placed onto the JES spool. Then JES assumes responsibility for sending it to the printer.

The LPD server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E will be issued and the server will terminate.

Considerations for using LPR/LPD

The LPR command parameters provide only limited functionality. There is no queue management interface with the LPR function and it has a limited query capability. The LPD server function also does not provide a robust queue management interface. Because it immediately places print files that are destined to mainframe controlled printers onto the JES spool, the only user query interface it provides is the limited LPQ client command, which enables the client to query the status of print jobs. However, it very seldom has anything to show the client. Once the file is under JES control, the LPD considers the file as having been delivered.

5.2.2 Configuration tasks for LPR/LPD

Use Table 5-1 to determine what print function you need to customize based on the origin of the print request. TSO interactive users and TSO batch jobs can use the LPR client function (LPR command) to print on IP network printers. Set up the LPD server on your z/OS system by creating an LPD configuration data set to enable remote LPR clients to use printers on your local JES system and Network Job Entry (NJE) network.

Table 5-1 LPR and LPD function table

Origin	Destination	Function
TSO user or batch Job	TCP/IP network printer	LPR
TCP/IP network client	z/OS JES or NJE printer	LPD

The line printer daemon (LPD) is the printer server that enables other hosts in your TCP/IP network to use printers on your z/OS system. You start the LPD server as an address space in your local system. The LPD server enables users in your TCP/IP network to address JES-controlled printers. A client from any TCP/IP host can use the local line printer requester (LPR) command to print a local file on a JES-controlled printer. The printer may be a local JES system printer, or it may be a printer accessed through an NJE network.

The LPR command enables a user on the local host to submit a file for printing on a remote printer server. Support is included for proper translation of carriage control information if the

file you want to print uses formatted carriage control characters in the file's records. If you have a PostScript® file on the z/OS system, you can use the LPR command to print that file on a PostScript printer in the TCP/IP network. *z/OS Communications Server: IP Configuration Guide*, SC31-8775, has a comprehensive chapter on setting up the LPD print server. The checklist in that chapter should be used to accomplish successful setup.

The configuration tasks for the LPD server are:

- Update TCP/IP Profile data set with AUTOLOG and PORT statements
- Customize the LPSERVE proc from SEZAINST(LPSPROC)
- Update LPD server configuration file from SEZAINST(LPDDATA)

Update TCP/IP Profile data set with AUTOLOG and PORT statements

An example of the AUTOLOG and PORT statements in the PROFILE data set is shown in Example 5-1.

Example 5-1 AUTOLOG and PORT statements for LPSERVE

```
AUTOLOG
   LPSERVEB
ENDAUTOLOG

PORT 515 TCP LPSERVEB
```

Update the LPD server cataloged procedure shipped as SYS1.SEZAINST(LPSPROC), but with an actual procedure name of LPSERVE.

Customize the LPSERVE proc from SEZAINST(LPSPROC)

An example of the cataloged procedure for LPSERVE is shown in Example 5-2.

Example 5-2 Cataloged procedure for LPSERVE

```
//LPSERVEB PROC MODULE=LPD,
//         LPDDATA=TCPIPB.TCPPARMS(LPDATB&SYSCLONE.),
//         LPDPRFX='PREFIX TCPIP',
//         DIAG=''
//*
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//LPD   EXEC PGM=MVPMAIN,
//  PARM=('&MODULE,ERRFILE(SYSERR),HEAP(512)',
//       'NOSPIE/ ''&LPDDATA'' &LPDPRFX &DIAG'),
//       REGION=6M,TIME=1440
//SPOOL OUTPUT CHARS=GT12
//STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR
//LPD1 OUTPUT CHARS=GT12
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.)
```

Update LPD server configuration file from SEZAINST(LPDDATA)

An example of the configuration data set, LPDDATA, for LPSERVE is shown in Example 5-3.

Example 5-3 The LPDDATA configuration data set for LPSERVE

```
BROWSE    TCPIPB.TCPPARMS(LPDATB31) - 01.00          Line 00000000 Col 001 080
;LPD CONFIGURATION DATA SET
;=========================
;
;DEBUG
;
 SERVICE sysprt1 PRINTER
    LOCAL
    FILTERS f l p r
    LINESIZE 132
    PAGESIZE 60
;
 SERVICE njeprt1 PRINTER
  NJE DEST=BRANCH22 IDENTIFIER=RMT2 OUTPUT=LPD1
  FILTERS f l p r
  LINESIZE 132
  PAGESIZE 60
;
 SERVICE lanprt4 PRINTER
  REMOTE pokip1145@lanprt.itso.ibm.com
; FAILEDJOB MAIL
;
SERVICE pun1 PUNCH
   LOCAL
   FILTERS l
   LINESIZE 80
;
;
OBEY CS07 CS01
```

5.2.3 Activation and verification of LPR/LPD

The startup messages for LPSERVE are shown in Example 5-4.

Example 5-4 LPSERVEB start up messages

```
S LPSERVEB
$HASP100 LPSERVEB ON STCINRDR
IEF695I START LPSERVEB WITH JOBNAME LPSERVEB IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 LPSERVEB STARTED
EZY1876I LPD STACK FUNCTIONS STARTED WITH PARAMETER LPD,ERRFILE(SYSERR
),HEAP(512),NOSPIE/ 'TCPIPB.TCPPARMS(LPDATB31)' PREFIX TCPIP .
```

A display of the TCP/IP stack connections shows the LPSERVE listener in Example 5-5.

Example 5-5 LPSERVEB listening on port 515

```
SDSF ULOG  CONSOLE CS07                          LINE 164     COLUMNS 44- 123
 COMMAND INPUT ===>                                           SCROLL ===> CSR
-d tcpip,tcpipb,n,conn
. . .
USER ID  CONN     STATE
 DCD1DIST 00004244 LISTEN
   LOCAL SOCKET:   0.0.0.0..38241
   FOREIGN SOCKET: 0.0.0.0..0
 DCD1DIST 00004241 LISTEN
   LOCAL SOCKET:   0.0.0.0..38240
```

```
   FOREIGN SOCKET: 0.0.0.0..0
 LDAPDBCG 00003096 LISTEN
   LOCAL SOCKET:   0.0.0.0..33389
   FOREIGN SOCKET: 0.0.0.0..0
 LPSERVEB 00004394 LISTEN
   LOCAL SOCKET:   0.0.0.0..515
   FOREIGN SOCKET: 0.0.0.0..0
. . .
```

In the following command discussions, the *printer* name is a printer defined on the *host* at the indicated location. The location can be either the IP address or the DNS name of the print server for the printer, and can be any print server in the network: either a z/OS system where LPSERVE is running as a started task, or a non-z/OS print server platform in the network.

The TSO LPR-related commands are:

- LPQ: used to query print job and queue status on a remote LPD server
- LPRM: used to remove one or more print jobs from the queue on a remote LPD server
- LPR: used to send a file to a remote LPD server to be printed
- LPRSET: used to set a default destination printer and print server, or to show the default

The LPQ command can query the status for all jobs on the printer's queue, ask for detailed status information related to those jobs, or query the status of jobs on the printer queue for a specific user ID or job number, as shown in Example 5-6.

Example 5-6 LPQ samples

```
LPQ (ALL PRINTER sysprt1 AT 10.1.1.20
LPQ (PRINTER lanprt4 AT wtsc31b.itso.ibm.com TRACE
LPQ myuserid (PRINTER nejprt1 AT 10.1.1.20
LPQ 273 (PRINTER lanprt4 AT wtsc31b.itso.ibm.com
```

See the *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, for a complete list of parameters that can be specified on the LPR command. Samples of the LPR command are shown in Example 5-7.

Example 5-7 LPR samples

```
LPR 'TESTFILE.PRINT' (PRINTER testprt1 AT lpdsrvr1 CLASS v COPIES 2 FILTER f l p r
                   JOB DEST=njenode3,IDENTIFIER=rmt23,FOR=myuserid

LPR 'TESTFILE.PRINT' (PRINTER testprt1 AT lpdsrvr1
```

The LPRM command can remove all jobs from the printer's queue, or just a specific job number, or a specific job name, as shown in Example 5-8.

Example 5-8 LPRM samples

```
LPRM (PRINTER testprt1 AT lpdsrvr1
LPRM 273 (PRINTER testprt1 AT lpdsrvr1
LPRM myjob (PRINTER testprt1 AT lpdsrvr1
```

The LPRSET command can specify the print server's DNS name or its IP address, or query the current default setting, or display a confirmation of the setting, as shown in Example 5-9.

Example 5-9 LPRSET samples

```
LPRSET (QUERY
EZB1020I Your LPR printer is currently set to <unassigned> at <unassigned>.
```

```
LPRSET testprt1@lpdsrvr1.ibm.com (TYPE
EZB1023I Printer set to testprt1 at lpdsrvr1.ibm.com.

LPRSET (QUERY
EZB1020I Your LPR printer is currently set to testprt1 at lpdsrvr1.ibm.com.

LPRSET testprt1@192.167.22.4
LPRSET (QUERY
EZB1020I Your LPR printer is currently set to testprt1 at 192.167.22.4.
```

If LPRSET has been used to set the default printer and server, then the other commands do not have to specify the printer or server, as in Example 5-10.

Example 5-10 Abbreviated LPR commands use the default setting

```
LPQ
LPR 'TESTFILE.PRINT'
LPRM 273
LPRM myjob
```

5.3 Infoprint Server

This section provides an overview of IP printing with the Infoprint Server. The Infoprint Server is a program product for the z/OS platform that uses z/OS UNIX System Services. This product is the basis for a total print serving solution for the z/OS environment. It enables you to consolidate your print workload from many servers onto a central z/OS print server.

This section discusses the following topics:

- Description of the Infoprint Server
- Configuration of Infoprint Server

5.3.1 Description of the Infoprint Server

The Infoprint Server executes in the z/OS UNIX System Services environment and requires z/OS UNIX system planning efforts and system setup.

The Infoprint Server provides support for LAN and host printing using your z/OS system as a centralized print management platform for the printing needs of the entire enterprise. It works together with data stream transforms that other IBM products provide. Figure 5-3 shows how most of the components of Infoprint Server fit into your z/OS system. The components of Infoprint Server and the transform products are shaded. The Infoprint Server Transforms components include:

- z/OS UNIX Shell Transform commands
- Infoprint Server Transforms
- Coax support

Archived

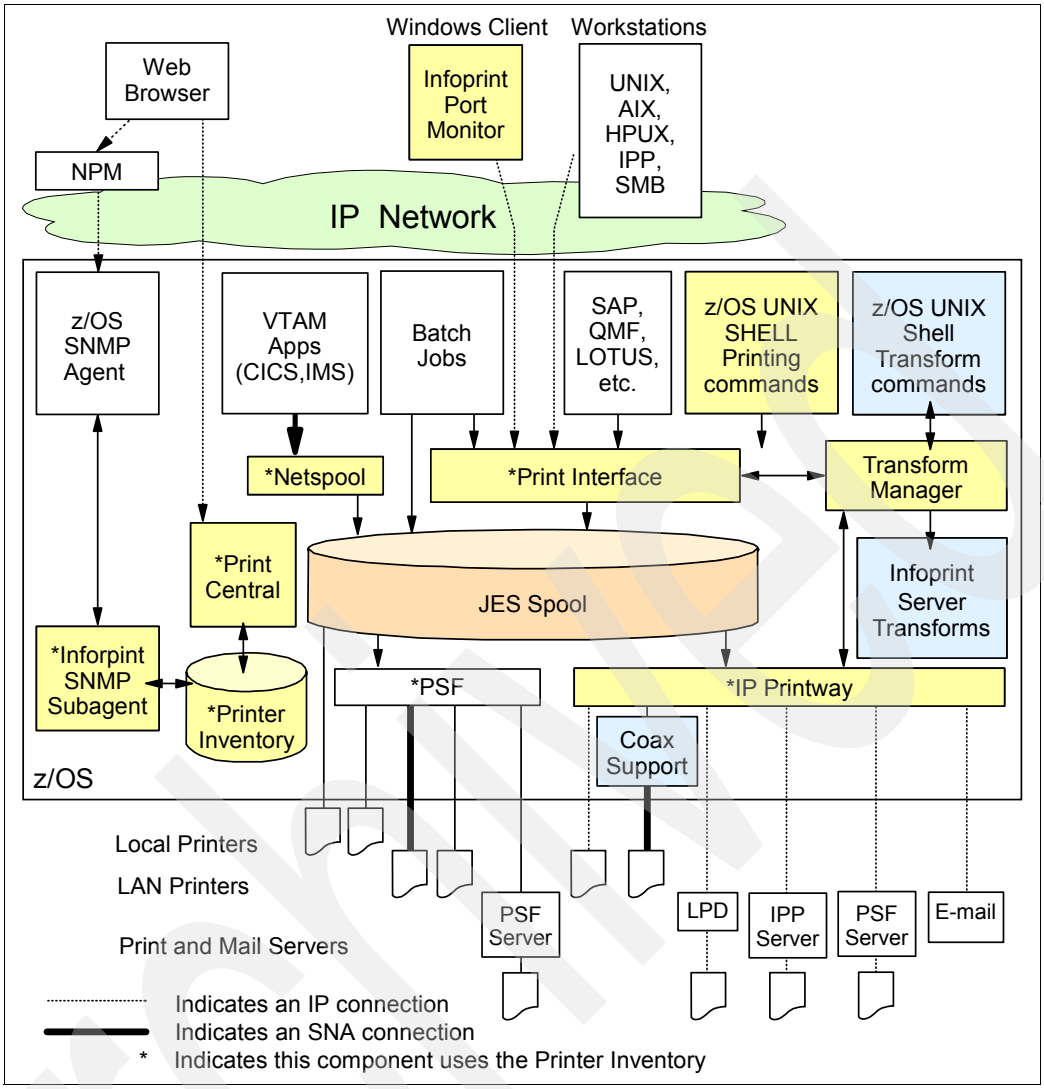


Figure 5-3 Infoprint components

Components of the Infoprint Server

The components of the Infoprint Server are described in this section.

Printer Inventory and Printer Inventory Manager

The Printer Inventory Manager controls the Printer Inventory. The Printer Inventory consists of files in the z/OS UNIX file system (zFS) that contain information about each printer and e-mail destination. The Printer Inventory also contains system configuration information for IP PrintWay™ and Print Services Facility™ (PSF) for z/OS.

Infoprint Server Windows Client

The Infoprint Server Windows client consists of the Infoprint Port Monitor, which sends print requests and job attributes to the Print Interface.

Print Interface

The Print Interface processes print requests from remote clients and from the local z/OS system and allocates output data sets on the JES spool. The Print Interface accepts various data formats and can transform input data streams to EBCDIC line data, ASCII text data, AFP,

PCL, PostScript, PDF, or other data formats that the printer accepts. A separate transform product is required for some transforms.

NetSpool

NetSpool™ processes print requests from VTAM applications, such as CICS and IMS™, and allocates output data sets on the JES spool. NetSpool thereby enables SNA applications to print to TCP/IP printers. NetSpool accepts SCS, 3270, and binary data streams, and can transform input data streams to EBCDIC line data, PCL, PDF, AFP, or other data formats that the printer accepts. A separate transform product is required for some transforms. However, a separate transform product is not required to convert input data streams to the line or PCL formats.

IP PrintWay

IP PrintWay transmits data sets from the JES spool to printers or print servers in a TCP/IP or SNA network and to e-mail destinations. IP PrintWay accepts various data formats and can transform input data streams to ASCII text data, PCL, PostScript, PDF, or other data formats that the printer accepts. A separate transform product is required for some transforms.

Transform Manager

The Infoprint Server Transform Manager manages transforms provided by Infoprint Server Transforms and other IBM transform products.

Infoprint Central

Infoprint Central is a Web-based application that lets help desk operators work with print jobs (output data sets) on the JES spool, printers controlled by IP PrintWay extended mode or PSF, and NetSpool logical units. It also lets operators see system status and printer definitions.

Simple Network Management Protocol (SNMP) subagent

The SNMP subagent lets you use an SNMP manager to view printer characteristics and printer status for printers that PSF controls and that do not have internal SNMP agents or that are not TCP/IP-attached to PSF. IBM Network Printer Manager for the Web (NPM), which you can download at no charge from the Web, enables an operator to monitor printers throughout the network from a Web browser running on any workstation.

Infoprint Server Transforms and other transforms (separate products)

IBM provides products that transform data streams from one format to another. These products are separate from the Infoprint Server.

PSF for z/OS (separate product)

The Print Services Facility (PSF) prints output on IBM AFP printers. The PSF system programmer can specify printer configuration information in the Printer Inventory for PSF to use when it starts a printer.

Advantages of using the Infoprint Server

The Infoprint Server delivers improved efficiency and lower overall printing cost with the flexibility for high-volume, high-speed printing from anywhere in the network. With the Infoprint Server, you can reduce the overall cost of printing while improving the management of output resources. Other benefits of the Infoprint Server are:

- IP PrintWay can give you fast access to TCP/IP-connected printers and to Virtual Telecommunications Access Method (VTAM)-controlled printers.

- NetSpool automatically directs VTAM application data to the Job Entry Subsystem (JES) spool without requiring application changes, enabling SNA print to be directed to IP printers.
- Infoprint Central lets help desk operators and other authorized users or job submitters work with print jobs, printers, and NetSpool logical units (LUs); display printer definitions; and check system status. Infoprint Central is a Web-based print management system.
- Infoprint Server Transforms, a separate product, provides a set of data transforms that lets you convert data to and from the AFP data format.
- IBM Infoprint XML Extender for z/OS, a separate product, lets you transform Extensible Markup Language (XML) files to AFP or PDF format for printing or e-mailing.
- IBM Infoprint XT Extender for z/OS, a separate product, lets you transform Xerox files to AFP format for printing or e-mailing. The Xerox files can be line-conditioned data streams (LCDS) or metacode data streams. XT is the IBM Xerox Transform technology.

Considerations for using the Infoprint Server

Even though the Infoprint Server is a separate product, it is comprised of a number of optional features, each of which requires its own configuration and planning.

The optional features of the Infoprint Server have interdependencies for each other, and the skills required to implement the product and its various features will probably cross departmental boundaries. A cooperative effort is required among all departments involved in order to achieve a successful implementation.

5.3.2 Configuration of Infoprint Server

Because the Infoprint Server is a separate product from the z/OS Communications Server, it is not within the scope of this book to cover the complete details of an Infoprint Server configuration. We list the major steps to be considered and then refer you to the reference material for the Infoprint Server product in “Additional information sources for IP printing” on page 183.

As part of any implementation effort two appendixes in this book should be beneficial in planning your work:

- Environment variables are categorized by application in Appendix A, “Environment variables” on page 311.

Sample files for each application are listed in Appendix B, “Sample files provided with TCP/IP” on page 323. Because we cannot cover the detailed implementation of the Infoprint Server in this book, refer to the reference material listed in 5.3, “Infoprint Server” on page 167 for implementation details.

A high-level overview of Infoprint Server implementation follows:

- Customizing the Infoprint Server components
- Operating Infoprint Server
- Administering the Infoprint Server
- Sample Infoprint Server configuration files for minimal functionality
- Sample Infoprint ISPF panels

Customizing the Infoprint Server components

This section can help you determine which Infoprint Server components you must customize to use the different functions that the Infoprint Server provides. Table 5-2 lists the functions

that the Infoprint Server provides and the Infoprint Server components you need to customize to support those functions.

Table 5-2 Infoprint functions with corresponding components

Infoprint Server function	Components
Receive print requests from these sources, and allocate output data sets on the JES spool: ► Clients that use LPR to LPD protocol ► Clients that use Internet Printing Protocol (IPP) ► Windows clients that use Server Message Block (SMB) protocol ► z/OS UNIX **`lp`**, **`lpstat`**, and **`cancel`** commands ► The AOPPRINT JCL procedure ► Any Windows application that supports printing ► Infoprint Server Application Programming Interface ► Batch jobs that specify the Print Interface subsystem on a DD statement	► Printer Inventory Manager ► Print Interface ► Infoprint Port Monitor for Windows (optional)
Receive print requests from VTAM applications (such as CICS and IMS), and allocate output data sets on the JES spool.	► Printer Inventory Manager ► NetSpool
Select output data sets from the JES spool and send data to a remote system using one of these transmission protocols: ► LPR to LPD protocol ► Internet Printing Protocol (IPP) ► Direct sockets printing ► VTAM ► E-mail	► Printer Inventory Manager ► IP PrintWay, basic or extended mode ► Print Interface (required to transform data when you use the resubmit for filtering function of IP PrintWay basic mode)
Transform data from one format to another, either automatically or with a z/OS UNIX transform command: afp2pcl, afp2pdf, afp2ps, pcl2afp, ps2afp, pdf2afp, sap2afp.	► Printer Inventory Manager ► Transform Manager ► Infoprint Server Transforms V1.1
Use Infoprint Central for the Web to work with print jobs, IP PrintWay extended mode printers, PSF printers, and NetSpool logical units.	► Printer Inventory Manager ► Infoprint Central
View printer characteristics and the status of PSF printers using an SNMP manager.	► Printer Inventory Manager ► SNMP subagent
Store PSF system information in the Printer Inventory.	► Printer Inventory Manager

Operating Infoprint Server

This section can help you determine which operational tasks you need to do to use the Infoprint Server components customized by your installation. Table 5-3 on page 171 lists the components of the Infoprint Server and the associated operational tasks.

Note: All components of the Infoprint Server require that you start the Printer Inventory Manager.

Table 5-3 Infoprint components and operational tasks

Component	Tasks
Printer Inventory Manager	► Start and stop Infoprint Server daemons. ► View messages.

Component	Tasks
Print Interface	► Start and stop Infoprint Server daemons. ► Use Infoprint Central to manage print jobs. ► Work with output data sets on the JES spool. ► View messages.
NetSpool	► Start and stop Infoprint Server daemons. ► Start and stop the NetSpool task. ► Use Infoprint Central to manage print jobs and NetSpool LUs. ► Work with output data sets on the JES spool View messages.
IP PrintWay basic mode	► Start and stop IP PrintWay FSAs. ► Maintain the IP PrintWay transmission-queue. ► Start sendmail. ► Work with output data sets on the JES spool. ► View messages.
IP PrintWay extended mode	► Start and stop Infoprint Server daemons. ► Start sendmail. ► Use Infoprint Central to work with print jobs and printers. ► Work with output data sets on the JES spool. ► View messages.
Transform Manager and Infoprint transforms	► Start and stop Infoprint Server daemons. ► View messages.
Infoprint Central	► Start and stop Infoprint Server daemons. ► Use Infoprint Central.
SNMP subagent	► Start and stop Infoprint Server daemons. ► View messages.

Administering the Infoprint Server

This section can help you determine which administrative tasks you need to do to use the Infoprint Server components customized by your installation. Table 5-4 lists the components of the Infoprint Server and the associated administrative tasks.

Table 5-4 Infoprint components and administrative tasks

Component	Tasks
Printer Inventory Manager	► Plan the Printer Inventory. ► Use ISPF panels to manage the Printer Inventory. ► Use the Printer Inventory Definition Utility (PIDU) to manage the Printer Inventory.
Print Interface	► Plan printer definitions for the Print Interface.
Infoprint transforms	► Plan printer definitions for data transforms.
NetSpool	► Plan printer definitions and printer pool definitions for NetSpool Define NetSpool printer logical units (LUs) to VTAM.
IP PrintWay	► Plan printer definitions for IP PrintWay. Use SMF type 6 accounting record written by IP PrintWay.

Sample Infoprint Server configuration files for minimal functionality

The Infoprint Server works from a configuration file, as shown in Example 5-11.

Example 5-11 Basic aopd.conf with the snmp subagent implemented

```
#-----------------------------------------------------------------------------
# aopd.conf - Base with snmp subagent
#-----------------------------------------------------------------------------
lpd-port-number = 515
ipp-port-number = 631
base-directory  = /var/Printsrv
ascii-codepage  = ISO8859-1
ebcdic-codepage = IBM-1047
job-prefix      = PR
inventory       = AOPD
start-daemons   = { lpd }
snmp-community  = j02cmd27
```

Message control can be customized as shown in Example 5-12.

Example 5-12 Certain messages can be selected for syslog

```
# aopmsg.conf - Infoprint Server Message Configuration file
#-----------------------------------------------------------------------------
hardcopy-messages = list
hardcopy-message-list = {AOP3614I AOP3803E}
#hardcopy-messages = all
#hardcopy-messages = none
```

Sample Infoprint ISPF panels

The Infoprint Server provides an ISPF panel interface for creating configuration files, printer definition files, and interface options to the other Infoprint functional components. For more details and descriptions see *z/OS Infoprint Server Operation and Administration*, S544-5745.

Some sample panels are shown here:

Figure 5-4 shows the main ISPF panel for an IP PrintWay printer definition.

```
                    IP PrintWay Printer Definition

Printer definition name . _______________
Description . ____________________________________________________ (extend)
Location. . . ____________________________________________________ (extend)

                        Component name        Custom values
Section                 (enter to list)       (enter to customize)
Allocation           => _______________        => __
Processing           => _______________        => *
NetSpool options     => _______________        => __
NetSpool end-of-file => _______________        => __
IP PrintWay options  => _______________        => __
Protocol             => _______________        => *

_ Use DEST, CLASS, and FORMS for IP PrintWay printer selection
NetSpool LU name . ________   LU classes . . __  __  __  __  __  __   (extend)
```

Figure 5-4 ISPF IP PrintWay Printer Definition panel

Figure 5-5 shows the main ISPF panel for a PSF printer definition.

```
                        PSF Printer Definition

 Printer definition name . _______________
 Description . ____________________________________________________ (extend)
 Location. . . ____________________________________________________ (extend)

                          Component name        Custom values
 Section                  (enter to list)       (enter to customize)
 Allocation            => _______________         => __
 Processing            => _______________         => *
 NetSpool options      => _______________         => __
 NetSpool end-of-file  => _______________         => __

 NetSpool LU name . _______    LU classes . . __  __  __  __  __  __   (extend)
```

Figure 5-5 ISPF PSF Printer Definition panel

Figure 5-6 shows the main ISPF panel for a General printer definition.

```
                        General Printer Definition

 Printer definition name . _______________
 Description . ____________________________________________________ (extend)
 Location. . . ____________________________________________________ (extend)

                          Component name        Custom values
 Section                  (enter to list)       (enter to customize)
 Allocation            => _______________         => __
 Processing            => _______________         => *
 NetSpool options      => _______________         => __
 NetSpool end-of-file  => _______________         => __
 IP PrintWay options   => _______________         => __

 NetSpool LU name . _______    LU classes . . __  __  __  __  __  __   (extend)
 Spooling mode. . . _   1. Line  2. Stream
```

Figure 5-6 ISPF General Printer Definition panel

Figure 5-7 shows the ISPF panel for the NetSpool Options component.

```
                              NetSpool Options

 Printer definition name . _______________


 Formatting . . . . 2   1. None  2. Convert to Line  3. Convert to PCL
    Record size . . ____
    RECFM . . . . . _   1. VB  2. VBA  3. VBM

 Default owner. . . ________
 Embedded attributes prefix . . ________________________
```

Figure 5-7 ISPF NetSpool Options panel

Figure 5-8 shows the ISPF panel for the IP PrintWay Options section or component.

```
                         IP PrintWay Options

Retention period:
   Successful. . . . _________   Failure . . _________
Retry time . . . . . _________
Retry limit. . . . . ____

Connection timeout . 30
Response timeout . . 600
Exits:
   Begin data set. . _______   End data set. . _______   Record. . _______

Document header  . . ____________________________________________(extend)
   / Translate document header
Document trailer . . ____________________________________________(extend)
   / Translate document trailer
_ Automatic dataset grouping (extended mode)
Dataset grouping. . . . 2  1. None  2. Job  3. Concatenate job

Formatting:
   Transparent data char . 35
   Delete form feed. . . . 1  1. None  2. Leading  3. Trailing  4. Both
   Carriage control type . _  1. None  2. Machine  3. ANSI
   _ Omit line termination at EOF

Basic Mode Formatting:
   Line termination. . . . _______
   Formatting. . . . . . . _  1. None            2. Standard
                              3. Translate only  4. Use FCB
   PostScript header . . . _  1. Add             2. Ignore
                              3. Landscape       4. Always landscape
```

Figure 5-8 ISPF IP PrintWay Options panel

Figure 5-9 shows the LPR Protocol panel.

```
                              LPR Protocol

Printer definition name . ______________
Operator security profile
   . . . ____________________________________________________

Printer IP address . ____________________________________________(extend)
Print queue name . . ____________________________________________(extend)

LPR Processing Options:
   Mode . . . . . . . . 2  1. Control file first  2. Control file last
                           3. Stream             4. Remote PSF
   _ Optimize copies
   _ Restrict ports
   / Print banner page
        Banner class. . ___________________________
        Banner job name ________________________________________(extend)
   Filename . . . . . . ___________________________
   Indent . . . . . . . ________
   Owner. . . . . . . . ___________________________
   Print function . . . f
   Title. . . . . . . . ________________________________________(extend)
   Width. . . . . . . . ________
   User options . . . . ________________________________________(extend)
```

Figure 5-9 ISPF LPR Protocol panel

Figure 5-10 shows the VTAM Protocol panel.

Archived

```
                              VTAM Protocol

Printer definition name . ______________
Operator security profile
   . . . ____________________________________________________________

Printer LU name. . . _______

VTAM Processing Options:
   Printer logmode. . . _______
   Checkpoint pages . . 5
   _ Send as transparent data
```

Figure 5-10 ISPF VTAM Protocol panel

Figure 5-11 shows the E-mail Protocol panel.

```
                              E-mail Protocol

Printer definition name . ______________

To addresses
   . . . ____________________________________________________ (more)
CC addresses
   . . . ____________________________________________________ (more)
BCC addresses
   . . . ____________________________________________________ (more)
From name . . .__________________________________________________
Reply address .__________________________________________________
```

Figure 5-11 ISPF E-mail Protocol panel

Figure 5-12 shows the ISPF panel for an IP PrintWay FSS definition.

```
                              IP PrintWay FSS

FSS name. . . _______
Description . ______________________________________________(extend)

   _ Old-style translation
  Hiperspace blocks . . ____
  TCP/IP job name . . . _______
  Document code page. . ____________
  Applid. . . . . . . . _______
  National language . . 1   1. English  2. Japanese
  Trace mode. . . . . . 1   1. None  2. Internal  3. No printing  4. Full
     __ Trace prompt
     Trace table size . ___
```

Figure 5-12 ISPF FSS printer definition panel

For complete implementation details on the Infoprint Server, refer to 5.3, “Infoprint Server” on page 167.

5.4 Problem determination for LPR/LPD

With z/OS Communications Server, authorization checks for many server applications are done at program startup, with an abend 4C5, reason code 77A53217, taken if the program is not properly authorized, as shown here:

```
IEA995I SYMPTOM DUMP OUTPUT
SYSTEM COMPLETION CODE=4C5  REASON CODE=77A53217
```

If an authorized program is to be started as a TSO command, it must be listed in the AUTHCMD section of IKJTSOxx, or TSO will consider it unauthorized, even if it resides in an authorized library.

See this techdoc for the details :
http://www-1.ibm.com/support/docview.wss?rs=852&context=SSSN3L&dc=DB520&dc=D600&dc=DB530&dc=D700&dc=DB500&dc=DB540&dc=DB510&dc=DB550&q1=77A53217&uid=swg21246822&loc=en_US&cs=utf-8&lang=en.

The LPR client command can fail if any of the following error conditions occur:

- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP may not allow the LPR client program access to the stack.
  - LPD may not be configured, or may be configured incorrectly.
- There is no LPD server listening on the remote server.
  - Ensure that you are using the correct port number.
  - Ensure that you are using the correct server name and IP address.

The TSO SMSG command provides an interactive interface to the LPD server to:

- Turn on and off diagnostics tracing.
- Query the work queue currently being used by the LPD server.

Example 5-13 shows the use of the SMSG command to turn the LPD TRACE on and off, and to show the work queue.

Example 5-13 SMSG being used to turn LPD TRACE on and off, and Print Queue

```
From the TSO ISHELL command line:
   smsg lpserveb trace on
   smsg lpserveb trace off
   smsg lpserveb print work
```

These commands are privileged, so the commands are only accepted from users specified in the OBEY statement in the LPD server configuration data set. Responses to the SMSG command are not sent to your TSO screen. You must look in the SYSPRINT file associated with the LPSERVE job to see the responses, as shown in Example 5-14.

Example 5-14 LPSERVE SMSG command responses

```
SDSF OUTPUT DISPLAY LPSERVEB STC07362  DSID   103 LINE 0       COLUMNS 02- 81
. . .
12:47:30 EZB0857I Using hardcoded translation tables
12:47:50 EZB0705I 09/23/07 12:47:50
12:47:50 EZB0834I Ready
14:55:03 EZB0786I Command received "TRACE ON".
14:55:03 EZB0789I GetNextNote with ShouldWait of TRUE
14:55:09 EZB0790I GetNextNote returns.  Connection 0 Notification SMSG received
14:55:09 EZB0786I Command received "TRACE OFF".
14:55:21 EZB0786I Command received "PRINT WORK".
```

```
14:55:21 EZB0731I       Work Queue start
14:55:21 EZB0733I       Work Queue end
```

On the TSO ISPF shell command line we issued the LPR command:

```
lpr 'tcpipb.tcpparms(databb30)' at 10.1.1.20 printer sysprt1 class h
```

We received the error message shown in Example 5-15 from the LPR command.

Example 5-15 Error message EZB0943I issued on the LPR command

```
EZB0943I No local printer ports available now.  Bind Conn failed.  Return Code = -1.
Error Number = 13.  Port Number = 731.   Remote IP Addr = 10.1.1.20
 ***
```

We have RESTRICTLOWPORTS specified in the IPCONFIG section of the stack's profile member. This includes the ports that LPR wants to use. We displayed the port reservation list using the netstat PORTLIST command and noticed that we had forgotten to reserve the ports for LPR, as seen in Example 5-16.

Example 5-16 Port reservation list indicates LPR ports missing (722-731)

```
d tcpip,tcpipb,n,portlist
. . .
PORT# PROT USER     FLAGS    RANGE       SAF NAME
00007 TCP  MISCSRVB DA
00009 TCP  MISCSRVB DA
00019 TCP  MISCSRVB DA
00020 TCP  OMVS
00021 TCP  OMVS     DA
00023 TCP  OMVS     DABU
      BINDSPECIFIC: 10.1.9.21
00023 TCP  TN3270B  DU
00025 TCP  SMTPB    DA
00053 TCP  NAMED9B  DA
00111 TCP  PORTMAPB DA
00512 TCP  OMVS     DABU
      BINDSPECIFIC: 10.1.9.22
00512 TCP  RXSERVEB DAU
00514 TCP  RXSERVEB DAU
00514 TCP  OMVS     DABU
      BINDSPECIFIC: 10.1.9.22
00515 TCP  LPSERVEB DA
00750 TCP  MVSKERBB DA
00751 TCP  ADM@SRVB DA
00992 TCP  TN3270B  D
00007 UDP  MISCSRVB DA
00009 UDP  MISCSRVB DA
00019 UDP  MISCSRVB DA
00053 UDP  NAMED9B  DA
00111 UDP  PORTMAPB DA
00161 UDP  SNMPDB   DA
00162 UDP  SNMPQEB  DA
00520 UDP  OMPB     D
00750 UDP  MVSKERBB DA
00751 UDP  ADM@SRVB DA
28 OF 28 RECORDS DISPLAYED
END OF THE REPORT
```

TSO users and batch jobs can issue LPR commands. Because we cannot know ahead of time what job names might be used for any TSO user ID or for batch jobs, we decided to use the wildcard approach to reserving the LPR ports. Because all of our user IDs started with CSxx, we set up an obeyfile member that would reserve the LPR ports for any job name starting with CS*, as shown in Example 5-17.

Example 5-17 OBEYFILE member showing LPR ports for LPR to be added to port reservation list

```
BROWSE    TCPIPB.TCPPARMS(LPRPORTS) - 01.00          Line 00000000 Col 001 080
PORT
 722 TCP CS*
 723 TCP CS*
 724 TCP CS*
 725 TCP CS*
 726 TCP CS*
 727 TCP CS*
 728 TCP CS*
 729 TCP CS*
 730 TCP CS*
 731 TCP CS*
```

We issued the OBEYFILE command to update the port list and then re-displayed the port reservation list to verify the LPR ports were added, as seen in Example 5-18 on page 179.

Note: When adding definitions to your stack's profile using the OBEYFILE command, if you want them to be permanent, do not forget to add the statements to the actual profile source before the next recycle of the stack. If they are not added to the source by the time the stack is recycled, the statements added by OBEYFILE will be lost.

Example 5-18 Port reservation list updated with LPR ports

```
-v tcpip,tcpipb,o,tcpipb.tcpparms(lprports)
 EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPB,O,TCPIPB.TCPPARMS(LPR
 PORTS)
 EZZ0300I OPENED OBEYFILE FILE 'TCPIPB.TCPPARMS(LPRPORTS)'
 EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIPB.TCPPARMS(LPRPORTS)
 '
 EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIPB.TCPPARMS(LPRPO
 RTS)'
 EZZ0053I COMMAND VARY OBEY COMPLETED SUCCESSFULLY

-d tcpip,tcpipb,n,portlist
. . .
 PORT# PROT USER     FLAGS    RANGE       SAF NAME
 00007 TCP  MISCSRVB DA
 00009 TCP  MISCSRVB DA
 00019 TCP  MISCSRVB DA
 00020 TCP  OMVS
 00021 TCP  OMVS     DA
 00023 TCP  OMVS     DABU
       BINDSPECIFIC: 10.1.9.21
 00023 TCP  TN3270B  DU
 00025 TCP  SMTPB    DA
 00053 TCP  NAMED9B  DA
 00111 TCP  PORTMAPB DA
 00512 TCP  OMVS     DABU
       BINDSPECIFIC: 10.1.9.22
 00512 TCP  RXSERVEB DAU
 00514 TCP  RXSERVEB DAU
```

```
 00514 TCP  OMVS     DABU
       BINDSPECIFIC: 10.1.9.22
 00515 TCP  LPSERVEB DA
 00722 TCP  CS*      DA
 00723 TCP  CS*      DA
 00724 TCP  CS*      DA
 00725 TCP  CS*      DA
 00726 TCP  CS*      DA
 00727 TCP  CS*      DA
 00728 TCP  CS*      DA
 00729 TCP  CS*      DA
 00730 TCP  CS*      DA
 00731 TCP  CS*      DA
 00750 TCP  MVSKERBB DA
 00751 TCP  ADM@SRVB DA
 00992 TCP  TN3270B  D
 00007 UDP  MISCSRVB DA
 00009 UDP  MISCSRVB DA
 00019 UDP  MISCSRVB DA
 00053 UDP  NAMED9B  DA
 00111 UDP  PORTMAPB DA
 00161 UDP  SNMPDB   DA
 00162 UDP  SNMPQEB  DA
 00520 UDP  OMPB     D
 00750 UDP  MVSKERBB DA
 00751 UDP  ADM@SRVB DA
 38 OF 38 RECORDS DISPLAYED
 END OF THE REPORT
```

Now we are ready to try the LPR command again. A display of the JES output queue for the LOCAL system printers shows *no output* from LPSERVEB jobname *before* our LPR command, as seen in Example 5-19.

Example 5-19 JES output queue before the LPR command shows nothing from LPSERVEB

```
SDSF OUTPUT ALL CLASSES ALL FORMS     LINES 441,414    LINE 38-56 (56)
 COMMAND INPUT ===>                                          SCROLL ===> CSR
 NP   JOBNAME  JobID    Owner    Prty C Forms    Dest                 Tot-Rec
      SYSLOG   STC03794 +MASTER+   96 A STD      LOCAL                 95,521
      PAGTRACF JOB04094 CS08      144 H STD      LOCAL                     92
      PAGTRACF JOB04095 CS08      144 H STD      LOCAL                    230
      TRMDRACF JOB04110 CS09      144 H STD      LOCAL                    101
      CNMEUNIX STC02494 SYSPROG   144 R STD      LOCAL                     80
      TRMDSETP JOB03936 CS08      144 R STD      LOCAL                     79
      TRMDSETP JOB03937 CS08      144 R STD      LOCAL                     80
      TRMDRACF JOB03950 CS08      144 R STD      LOCAL                     21
      TRMDRACF JOB03951 CS08      144 R STD      LOCAL                     95
      TRMDRACF JOB04023 CS08      144 R STD      LOCAL                    101
 SDSF HELD OUTPUT DISPLAY ALL CLASSES  LINES 904        LINE 1-4 (4)
 COMMAND INPUT ===>                                          SCROLL ===> CSR
 NP   JOBNAME  JobID    Owner    Prty C ODisp Dest                 Tot-Rec  Tot-
      CS074    JOB07178 CS07      144 H HOLD  LOCAL                     46
      CS075    JOB07180 CS07      144 H HOLD  LOCAL                     46
      CS07     TSU07131 CS07      144 S HOLD  LOCAL                    276
      CS07     TSU07132 CS07      144 S HOLD  LOCAL                    536
```

This time we decided to turn on the LPD Trace using the TSO SMSG command before we re-issued the LPR command:

```
smsg lpserveb trace on
```

```
lpr 'tcpipb.tcpparms(databb30)' at 10.1.1.20 printer sysprt1 class h
```

We can now look at SYSPRINT for LPSERVEB for the trace output, and at the SDSF JES output queue for output from LPSERVEB in class H (specified on the LPR command). The trace output was quite large, so we do not show all of it here. However, we do include some important lines from the trace that show the progression of servicing the received print file from LPR. The trace is shown in Example 5-20.

Example 5-20 LPD trace receiving and processing a file from LPR

```
15:27:24 EZB0786I Command received "TRACE ON".
15:27:24 EZB0789I GetNextNote with ShouldWait of TRUE
16:22:05 EZB0790I GetNextNote returns.  Connection 0 Notification Connection sta
16:22:05 EZB0779I New connection state Open (8673) on connection 0 with reason 0
16:22:05 EZB0626I Allocated ConnectionBlock at 001C7E08
16:22:05 EZB0627I Passive open on port 515
..............................
16:22:05 EZB0716I Job 115 received sysprt1 10.1.1.20
16:22:05 EZB0734I Job 115 added to work queue
16:22:05 EZB0716I Job 115 scheduled sysprt1 10.1.1.20
16:22:05 EZB0776I Released StepBlock at 000060C8
16:22:05 EZB0777I Released ConnectionBlock at 001BFE08
16:22:05 EZB0824I ProcessWork starting on job queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBstartPRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0825I    Job 115 for sysprt1 dispatched in state JOBstartPRINTING
16:22:05 EZB0716I Job 115 printing sysprt1 10.1.1.20
16:22:05 EZB0827I ProcessWork end with queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBcontinuePRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0789I GetNextNote with ShouldWait of FALSE
16:22:05 EZB0824I ProcessWork starting on job queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBcontinuePRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0825I    Job 115 for sysprt1 dispatched in state JOBcontinuePRINTING
16:22:05 EZB0827I ProcessWork end with queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBfinishPRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0789I GetNextNote with ShouldWait of FALSE
16:22:05 EZB0824I ProcessWork starting on job queue
16:22:05 EZB0731I       Work Queue start
16:22:05 EZB0732I $        115 JOBfinishPRINTING
16:22:05 EZB0733I       Work Queue end
16:22:05 EZB0825I    Job 115 for sysprt1 dispatched in state JOBfinishPRINTING
16:22:05 EZB0716I Job 115 sent sysprt1 10.1.1.20
16:22:05 EZB0769I Job 115 removed from work queue
16:22:05 EZB0751I Released StepBlock at 00006280
16:22:06 EZB0716I Job 115 purged sysprt1 10.1.1.20
16:22:06 EZB0771I Released JobBlock at 000AE960
16:22:06 EZB0827I ProcessWork end with queue
16:22:06 EZB0731I       Work Queue start
16:22:06 EZB0733I       Work Queue end
16:22:06 EZB0789I GetNextNote with ShouldWait of TRUE
```

We turned the trace off and issued another LPR command, specifying a different member to be printed, this time:

```
lpr 'tcpipb.tcpparms(databb31)' at 10.1.1.20 printer sysprt1 class h
```

When the trace is off, the output in LPSERVEB's SYSPRINT appears as it does in Example 5-21.

Example 5-21 LPSERVEB normal processing messages when trace is off

```
16:35:12 EZB0716I Job 88 received sysprt1 10.1.1.20
16:35:12 EZB0716I Job 88 scheduled sysprt1 10.1.1.20
16:35:12 EZB0716I Job 88 printing sysprt1 10.1.1.20
16:35:12 EZB0716I Job 88 sent sysprt1 10.1.1.20
16:35:12 EZB0716I Job 88 purged sysprt1 10.1.1.20
```

We look at the JES output queue for output from LPSERVEB in *class H* (specified on the LPR command). We now see two new entries in the output queue for LPSERVEB that were not there before:

- One is for LPSERVEB's job 115 when the trace was on (DATAB30).
- The other is for LPSERVEB's job 88 when the trace was off (DATAB31).

These entries are seen in Example 5-22.

Example 5-22 LPSERVEB entries in the JES output queue

```
SDSF OUTPUT ALL CLASSES ALL FORMS     LINES 441,440    LINE 47-58 (58)
COMMAND INPUT ===>                                        SCROLL ===> CSR
NP   JOBNAME  JobID    Owner    Prty C Forms    Dest                Tot-Rec
     SYSLOG   STC03794 +MASTER+   96 A STD      LOCAL                95,521
     PAGTRACF JOB04094 CS08      144 H STD      LOCAL                    92
     PAGTRACF JOB04095 CS08      144 H STD      LOCAL                   230
     TRMDRACF JOB04110 CS09      144 H STD      LOCAL                   101
s    LPSERVEB STC07362 TCPIP     144 H STD      LOCAL                    13
s    LPSERVEB STC07362 TCPIP     144 H STD      LOCAL                    13
     CNMEUNIX STC02494 SYSPROG   144 R STD      LOCAL                    80
     TRMDSETP JOB03936 CS08      144 R STD      LOCAL                    79
     TRMDSETP JOB03937 CS08      144 R STD      LOCAL                    80
     TRMDRACF JOB03950 CS08      144 R STD      LOCAL                    21
     TRMDRACF JOB03951 CS08      144 R STD      LOCAL                    95
     TRMDRACF JOB04023 CS08      144 R STD      LOCAL                   101
```

If we select these two entries, we can verify that the DATAB30 and DATAB31 members are indeed in the queue, ready for printing on the local system printer, waiting in class H, as seen in Example 5-23.

Example 5-23 LPSERVEB contents of the two print files waiting to be printed

```
SDSF OUTPUT DISPLAY LPSERVEB STC07362  DSID   111 LINE 0       COLUMNS 02- 81
WTSC30B.ITSO.IBM.COM:TCPIPB.TCPPARMS(DATAB30)

TCPIPJOBNAME TCPIPB
HOSTNAME WTSC30B
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  10.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

SDSF OUTPUT DISPLAY LPSERVEB STC07362  DSID   112 LINE  DATA SET DISPLAYED
```

```
WTSC30B.ITSO.IBM.COM:TCPIPB.TCPPARMS(DATAB31)

TCPIPJOBNAME TCPIPB
HOSTNAME WTSC31B
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  10.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
```

5.5 Additional information sources for IP printing

Detailed information for IP printing can be found in the following documents:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- Infoprint Server migration chapter in *z/OS Migration*, GA22-7499
- *z/OS Infoprint Server Introduction*, S544-5742
- *z/OS Infoprint Server Customization*, S544-5744
- *z/OS Infoprint Server Operation and Administration*, S544-5745
- *z/OS Infoprint Server User's Guide*, S544-5746
- *z/OS Infoprint Server Implementation*, SG24- 6234
- LPR/LPD is defined by RFC 1179

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

Archived

Archived

6

INETD

INETD, also known as the Internet super daemon, is an application that listens for connection requests on behalf of other applications. The handling the initial connection process, INETD passes the connection to the application associated with the targeted port. This chapter focuses on INETD functions that are available in the z/OS Communications Server.

This chapter discusses the following INETD topics.

Section	Topic
6.1, “Conceptual overview of INETD” on page 186	The basic concepts of INETD.
6.2, “A single INETD setup” on page 188	How to configure INETD to listen on behalf of a number of dependent applications.
6.3, “Problem determination for INETD” on page 194	Problem determination techniques for INETD.
6.4, “Additional information sources for INETD” on page 195	References to additional reading resources for INETD topics.

© Copyright IBM Corp. 2008. All rights reserved.

6.1 Conceptual overview of INETD

INETD is one of the standard applications provided with the z/OS Communications Server. It is a generic listener than can be used by any server that does not have its own listener. The relationship between INETD and its supported applications is shown in Figure 6-1.

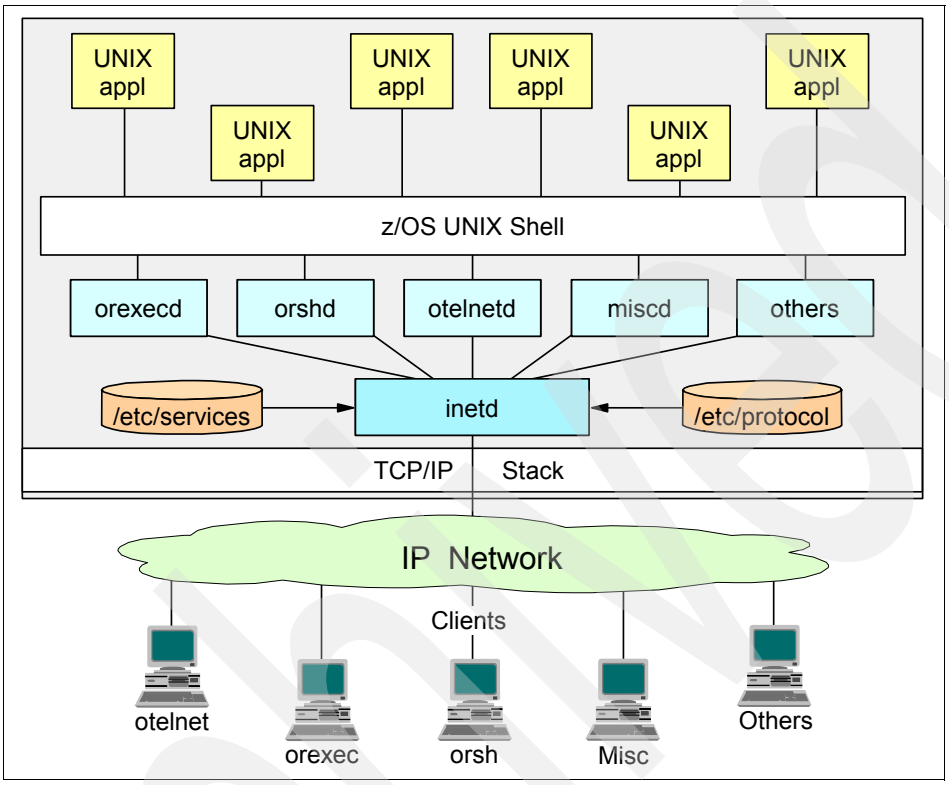


Figure 6-1 INETD and its supported applications

This section discusses the following concepts:

- What is INETD?
- How does INETD work?
- How can INETD be applied?

6.1.1 What is INETD?

The INETD servers provide access to the z/OS UNIX shell via otelnetd, rexecd, or rshd, allowing you to then run other UNIX commands and applications from there. The z/OS Communications Server includes three applications and a number of internal services that require INETD; these are listed in Table 6-1.

Table 6-1 Applications that use INETD

Application	Description
z/OS UNIX Telnet Server	Refer to Chapter 8, “z/OS UNIX Telnet server” on page 235.
z/OS UNIX REXEC Server	Refer to Chapter 9, “Remote execution” on page 245.
z/OS UNIX RSH server	Refer to Chapter 9, “Remote execution” on page 245.
echo	Repeats any data received back to the sender.

Application	Description
discard	Throws away any received data.
chargen	Sends predefined or random data and discards any received data.
daytime	Sends the current date and time in user readable form.
time	Sends the current date and time in machine readable form.
popper	Refer to Chapter 7, “z/OS mail servers” on page 197.

6.1.2 How does INETD work?

INETD calls fork() to run in the background and disassociates itself from the controlling terminal. z/OS UNIX appends a 1 to the job name, provided the job name is less than eight characters. For example, if INETD is started with jobname INETD, after the application forks the job name is INETD1. The correct job name of INETD is important to note because the TCPIP.PROFILE data set should reserve ports for INETD using the job name after INETD forks.

INETD can accept two command line parameters: -d and a file name. Both parameters are optional. If -d is specified, INETD does not fork() at startup and all error messages are written to STDERR. If a file name is specified, then INETD uses the specified file as the configuration file instead of the default /etc/inetd.conf.

Note: Only one INETD daemon can be run in one MVS image, the process file used by INETD daemon is /etc/inetd.pid, you can’t change it to another file name neither by shell commands or environment variables, and it only can be used by one INETD daemon at the same time.

6.1.3 How can INETD be applied?

INETD reduces system load by invoking other daemons only when they are needed and by providing several simple Internet services internally without invoking other daemons. Some applications depend on INETD to provide a listener, and those applications cannot be started without INETD.

The /etc/inetd configuration file

INETD uses a configuration file in the z/OS UNIX file system to determine for which services INETD will listen. A sample configuration file can be found in the /samples/inetd.conf file. The format of the file is:

```
<service> <socket type> <protocol> <wait/nowait> <user> <application> <arguments>
```

All parameters except the last parameter, arguments, are required on each line. Table 6-2 defines each parameter.

Table 6-2 Configuration file parameters

Parameter	Description
service	Name of the Internet service. This name must match an entry in /etc/services. By default, INETD assumes you want to listen on all IP addresses. However, you can optionally specify the service parameter in the format of IP_address:service_name to force a particular service to listen on a particular IP address.

Parameter	Description
socket type	Options are stream or dgram. Applications that use the TCP protocol are stream applications. Applications using UDP are dgram.
protocol	The IP protocol used by the application. The options are TCP, UDP, TCP4, TCP6, UDP4, or UDP6. If TCP6 or UDP6 is specified then the socket will support IPv6. You may also optionally specify the maximum receive buffer in the format of protocol,sndbuf=*n*, where *n* is the number of users.
wait/nowait	Wait indicates the server is single threaded and the application will issue an accept() API call itself and process connections one at a time. Nowait indicates the application is multi threaded. In nowait mode, INETD issues the accept() API call and passes a connected socket to the application. The application in nowait mode can process multiple connections at a time. You can also optionally specify the maximum number of simultaneous users allowed by the application by using the format nowait.*n* or wait.*n*, where *n* is the number of users.
user	The user ID the application should run under.
application	The full path to the executable file for the application INETD should start.
arguments	Up to 20 optional arguments that may be passed to the application.

6.2 A single INETD setup

INETD is a simple application with limited configuration options. INETD requires an active stack and an /etc/inetd.conf configuration file. We show a single design scenario for INETD. We recommend using INETD only if you want to start otelnetd, orexecd, orshd, or if you want to use the internal services of INETD for testing purposes. In our simple design we will configure INETD with all services active. If a particular service is not desired in your environment, simply comment out the statements for the undesired service.

6.2.1 Description of the INETD setup

In this section we discuss a the INET implementation using the more common configuration options. We show a configuration file that starts all servers and internal services. This design allows the INETD configuration to be easily changed when INETD is running. Simply comment out a line in the configuration file and send a SIGHUP signal to the INETD task to stop a service.

It is never a good idea to start unnecessary server applications that will not be used. Therefore, we recommend giving careful consideration to your INETD configuration file and only allowing services to be started that will be used. For example, if you have no need for the internal services provided by INETD, then comment out the associated lines for the internal services in the configuration file.

6.2.2 Configuration tasks for INETD setup

The tasks are:

- Create an INETD configuration file
- Update Port statements in /etc/services
- Reserve ports in PROFILE.TCPIP

Create an INETD configuration file

First, copy the sample configuration file from /samples/inetd.conf to a new location such as /etc/inetd.conf.

Next, examine the configuration file and determine which services you want to use. Add or comment out lines as necessary. In this example, we uncomment the lines in the sample and add additional lines for each server.

Finally, you will need to determine which user name INETD and the servers started by INETD will run under. We recommend using OMVSKERN. The user name should have read access to the BPX.DAEMON facility class in RACF.

Example 6-1 shows our INETD configuration file.

Example 6-1 INETD configuration file

```
###
# Internet server configuration database
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 2001
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# /etc/inetd.conf
#
#          Internet server configuration database
#
#  $01=PYQ0049,  HOT7705, 010130, PDJP: Correct paths and remove
#        unsupported services (FIN APAR OW45915
#
# Services can be added and deleted by deleting or inserting a
# comment character (i.e. #) at the beginning of a line
#
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
pop3     stream tcp nowait OMVSKERN /usr/sbin/popper popper
echo     stream tcp nowait OMVSKERN internal
discard  stream tcp nowait OMVSKERN internal
chargen  stream tcp nowait OMVSKERN internal
daytime  stream tcp nowait OMVSKERN internal
time     stream tcp nowait OMVSKERN internal
echo     dgram  udp wait   OMVSKERN internal
discard  dgram  udp wait   OMVSKERN internal
chargen  dgram  udp wait   OMVSKERN internal
daytime  dgram  udp wait   OMVSKERN internal
time     dgram  udp wait   OMVSKERN internal
```

Update Port statements in /etc/services

For each service in the INETD configuration file, make sure a matching entry is specified in the ETC.SERVICES data set or /etc/services file. Refer to our /etc/services file in Example 6-2.

Example 6-2 /etc/services

```
echo            7/tcp
echo            7/udp
discard         9/tcp
discard         9/udp
daytime         13/tcp
daytime         13/udp
chargen         19/tcp
chargen         19/udp
otelnet         23/tcp
time            37/tcp
time            37/udp
pop3            110/tcp
exec            512/tcp
login           513/tcp
shell           514/tcp
```

Reserve ports in PROFILE.TCPIP

While not required, we recommend that you reserve ports in PROFILE.TCPIP to the INETD job name. If you do not reserve ports, it is possible that some other application will bind to the ports normally reserved for INETD servers. Our PORT statement in PROFILE.TCPIP is shown in Example 6-3 and reserves the ports for job name INETD1.

Important: Remember that the job name of INETD changes after INETD is started.

Example 6-3 PORT statement from PROFILE.TCPIP

```
PORT
    7    TCP   INETD1
    7    UDP   INETD1
    9    TCP   INETD1
    9    UDP   INETD1
    13   TCP   INETD1
    13   UDP   INETD1
    19   TCP   INETD1
    19   UDP   INETD1
    23   TCP   INETD1 BIND 10.1.9.21
    37   TCP   INETD1
    37   UDP   INETD1
    110  TCP   INETD1
    512  TCP   INETD1 BIND 10.1.9.22
    513  TCP   INETD1
    514  TCP   INETD1 BIND 10.1.9.22
```

Note: Refer to Chapter 8, “z/OS UNIX Telnet server” on page 235, for details on why port 23 might be bound to a specific IP address.

6.2.3 Activation and verification of INETD

There are two methods for starting INETD: by shell script in /etc/rc or as a started task. We recommend starting the INETD daemon via a shell script in /etc/rc because causes INETD to be automatically started when z/OS UNIX is started.

The best method to verify that INETD is running correctly is to issue a **netstat** command to see the listener connections for INETD.

The activation and verification techniques for INETD and its listener processes are:

- Starting INETD
- Verifying INETD initial startup
- Verifying the echo service
- Verifying the discard service
- Verifying the daytime service
- Verifying the chargen service
- Verifying the time service

Starting INETD

A sample shell script is shown in Example 6-4. An example JCL procedure to start INETD is included in Example 6-5.

Example 6-4 Shell script commands in /etc/rc to start INETD

```
_BPX_JOBNAME='INETD'
/usr/sbin/inetd &
```

Example 6-5 JCL to start INETD

```
//INETD   PROC INETDENV=INETDENV
//INETD   EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//         PARM='PGM /usr/sbin/inetd /etc/inetd.conf'
//STDOUT   DD PATH='/tmp/inetd-stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDERR   DD PATH='/tmp/inetd-stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDENV   DD DSN=SYS1.TCPPARMS(&INETDENV.),DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=SYS1.TCPPARMS(DATA&SYSCLONE.)
```

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

RECFM=F must *not* be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in UNIX System Services. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Verifying INETD initial startup

Issue the TSO (NETSTAT) or UNIX (**`netstat`**) command, filtering on the client ID of INETD. If INETD is started correctly, the output of the **`netstat`** command should be similar to the output shown in Example 6-6.

```
NETSTAT CONN (CLIENT INETD*
   or
netstat -c -E INETD1
```

Example 6-6 Output of netstat after INETD is started

```
. . .
User Id  Conn     State
-------  ----     -----
```

```
INETD1   00000049 Listen
  Local Socket:   10.1.9.22..512
  Foreign Socket: 0.0.0.0..0
INETD1   00000046 Listen
  Local Socket:   0.0.0.0..19
  Foreign Socket: 0.0.0.0..0
INETD1   00000048 Listen
  Local Socket:   0.0.0.0..7
  Foreign Socket: 0.0.0.0..0
INETD1   00000045 Listen
  Local Socket:   0.0.0.0..13
  Foreign Socket: 0.0.0.0..0
INETD1   0000004C Listen
  Local Socket:   10.1.9.21..23
  Foreign Socket: 0.0.0.0..0
INETD1   00000047 Listen
  Local Socket:   0.0.0.0..9
  Foreign Socket: 0.0.0.0..0
INETD1   00000044 Listen
  Local Socket:   0.0.0.0..37
  Foreign Socket: 0.0.0.0..0
INETD1   0000004D Listen
  Local Socket:   0.0.0.0..110
  Foreign Socket: 0.0.0.0..0
INETD1   0000004B Listen
  Local Socket:   10.1.9.22..514
  Foreign Socket: 0.0.0.0..0
INETD1   0000004A Listen
  Local Socket:   0.0.0.0..513
  Foreign Socket: 0.0.0.0..0
INETD1   00000042 UDP
  Local Socket:   0.0.0.0..9
  Foreign Socket: *..*
INETD1   00000041 UDP
  Local Socket:   0.0.0.0..19
  Foreign Socket: *..*
INETD1   00000043 UDP
  Local Socket:   0.0.0.0..7
  Foreign Socket: *..*
INETD1   00000040 UDP
  Local Socket:   0.0.0.0..13
  Foreign Socket: *..*
INETD1   0000003F UDP
  Local Socket:   0.0.0.0..37
  Foreign Socket: *..*
```

Note that there are 15 entries associated with job name INETD1, which match the 15 services that were defined in the INETD configuration file. The netstat output only verifies that INETD is started and that it has *listeners* for each application. However, netstat does not verify that each of the servers is working.

We now verify that the internal services of INETD are working correctly by using the DOS Telnet client included in Microsoft® Windows to Telnet to each port and return the output. The remote execution servers (orexecd and orshd) are discussed in detail in Chapter 9, “Remote execution” on page 245, and otelnetd is discussed in Chapter 8, “z/OS UNIX Telnet server” on page 235.

Verifying the echo service

The echo service can be verified by using a Telnet client to Telnet to port 7 tcp, pressing Enter, and then slowly typing `This is a test`. If the echo service is working correctly, each letter is repeated as it is typed. We received the output shown in Figure 6-2.

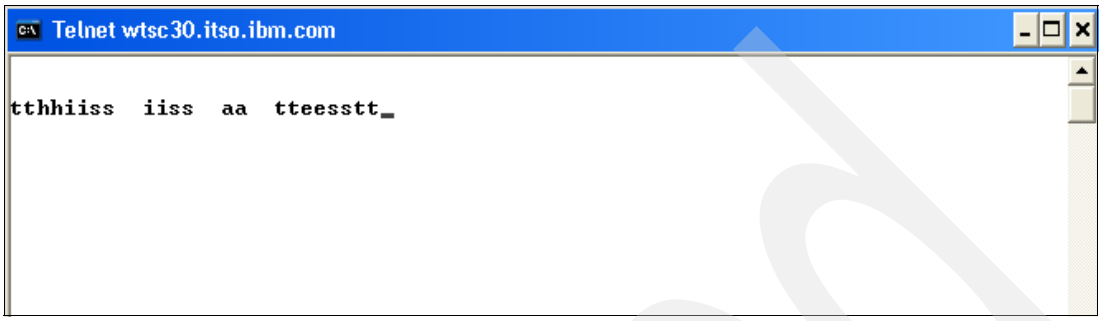


Figure 6-2 Verification of the echo service

Verifying the discard service

The discard service can be verified by using a Telnet client to Telnet to port 9 tcp, pressing Enter, and typing `This is a test`. If the echo service is working correctly, the curser moves, but each character is discarded as it is received. We received the output shown in Figure 6-3.

Figure 6-3 Verification of the discard service

Verifying the daytime service

The daytime service can be verified by using a Telnet client to Telnet to port 13 tcp and pressing Enter if needed. The daytime service responds by printing the human readable date and time string and then disconnecting. We received the output shown in Figure 6-4.

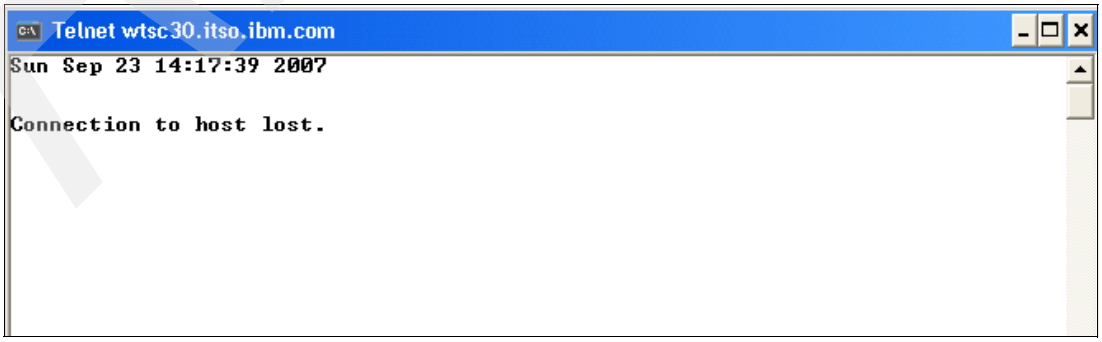


Figure 6-4 Verification of the daytime service

Archived

Verifying the chargen service

The chargen service can be verified by using a Telnet client to Telnet to port 19 tcp. The chargen service sends a known string of text until the client disconnects. We received the output shown in Figure 6-5 on page 194.

```
Telnet wtsc30.itso.ibm.com
.<(+|&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNO
<(+|&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOP
(+|&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQ
+|&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR
|&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\
&!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\S
!$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\ST
$*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STU
*);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUV
);^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVW
;^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWX
^-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXY
-/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ
/,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ0
,%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ01
%_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ012
_>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ0123
>?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ01234
?`:#@'="abcdefghijklmnopqr~stuvwxyz[]{ABCDEFGHI}JKLMNOPQR\STUVWXYZ012345
```

Figure 6-5 Verification of the chargen service

Verifying the time service

The time service can be verified by using a Telnet client to Telnet to port 37 tcp. The time service sends an unreadable binary string representing the time and date. We received the output shown in Figure 6-6.

```
Telnet wtsc30.itso.ibm.com
Connection to host lost.
```

Figure 6-6 Verification of the time service

6.3 Problem determination for INETD

To perform problem determination with INETD itself or with one of the internal servers, start INETD with the -d option to enable debugging.

> **Note:** The -d option prevents INETD from forking at startup. Therefore the job name does not change. Since INETD will not fork in debugging mode, the behavior of INETD may also change. Additionally, the -d option results in numerous messages sent to the STDERR data stream.

Refer to Chapter 9, “Remote execution” on page 245, and Chapter 8, “z/OS UNIX Telnet server” on page 235, for information regarding remote execution or otelnetd.

6.4 Additional information sources for INETD

For additional information, refer to:

- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- The INETD man page (Issue **`man inetd`** from the z/OS UNIX shell.)

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Archived

Archived

7

z/OS mail servers

This chapter focuses on the mail services that are available in the z/OS Communications Server. The basic Internet mail protocols provide mail (note) and message exchange between TCP/IP hosts. Facilities have been added for the transmission of data that cannot be represented as 7-bit ASCII text. There are three standard protocols that apply to mail of this kind. Each is recommended. The term Simple Mail Transfer Protocol (SMTP) is frequently used to refer to the combined set of protocols because they are so closely inter-related but, strictly speaking, SMTP is just one of the three. Normally, it is evident from the context which of the three protocols is being referenced. Whenever some doubt might exist, we refer to the STD or RFC numbers to avoid ambiguity.

This chapter discusses the following mail server topics.

Section	Topic
7.1, “Conceptual overview of z/OS mail applications” on page 198	The basic concepts of mail servers.
7.2, “z/OS SMTP as a mail server” on page 200	Commonly implemented mail server design for z/OS SMTP with configuration examples and verifications procedures.
7.3, “Sendmail and popper as mail servers” on page 213	Sendmail and popper configurations examples and verification procedures.
7.4, “Using sendmail as a client” on page 227	Sendmail client setup configuration examples and verification procedures.
7.5, “Problem determination for the mail facilities” on page 231	Problem determination tools and commands for z/OS mail services.
7.6, “Additional information sources for mail servers” on page 233	References to additional documentation for z/OS mail services.

© Copyright IBM Corp. 2008. All rights reserved.

7.1 Conceptual overview of z/OS mail applications

As illustrated in Figure 7-1, SMTP and sendmail are standard applications provided with the z/OS Communications Server. Both SMTP and sendmail use the Pascal API which communicates directly with the TCP transport layer of the TCP/IP stack. The Pascal API bypasses communication with the z/OS UNIX Logical and Physical file systems.

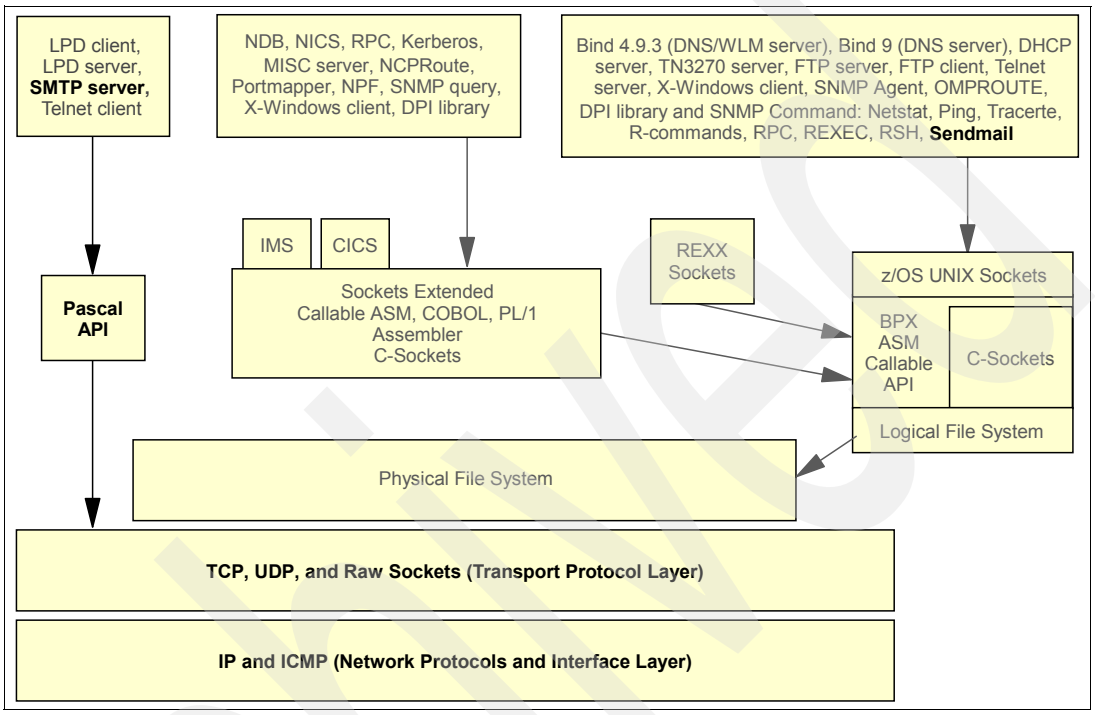


Figure 7-1 z/OS mail services

This section discusses the following topics:

- What are z/OS mail services?
- How do z/OS mail services work?
- How can z/OS mail services be applied?

7.1.1 What are z/OS mail services?

This section provides an overview of the TCP/IP application protocols dealing with electronic mail. Two mail functions are supported on z/OS:

- Simple Mail Transport Protocol (SMTP) in the MVS environment
- Sendmail as a z/OS UNIX component

The various standards that define the protocols for sending electronic mail are numerous. The Request For Comment (RFC) numbers that represent these standards and their terminology can be very confusing. We provide an overview of the main RFC numbers and their descriptions here.

RFC 821 defines a client/server protocol. As usual, the client SMTP is the one that initiates the session (that is, the sending SMTP), and the server is the one that responds (the receiving SMTP) to the session request. However, because the client SMTP frequently acts as a server for a user mailing program, it is often simpler to refer to the client as the sender SMTP and to the server as the receiver SMTP.

7.1.2 How do z/OS mail services work?

The relationship of the z/OS SMTP server and the client network is illustrated in Figure 7-2. The users are clients, entering client commands or requesting other mail services.

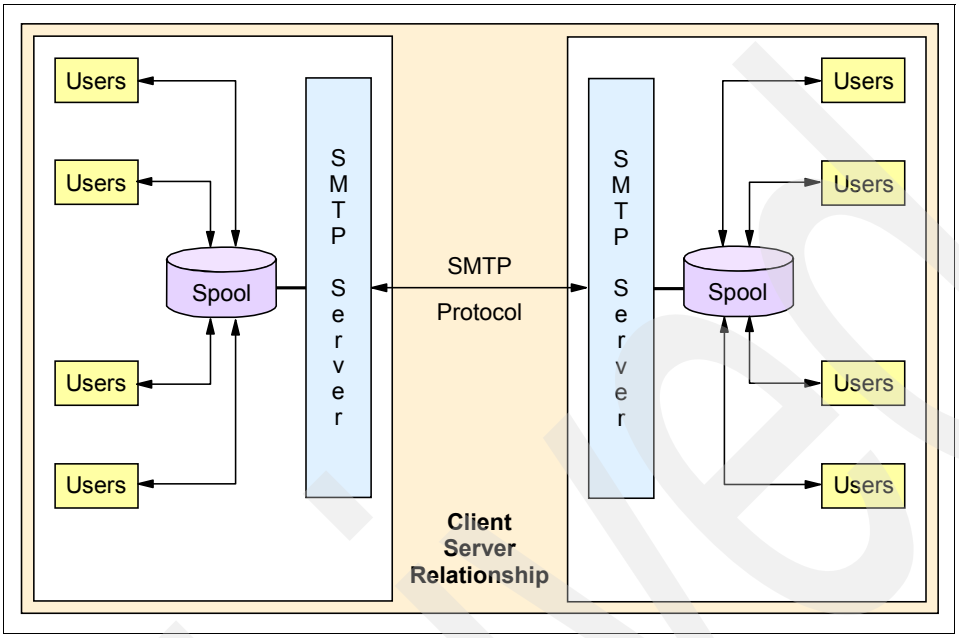


Figure 7-2 SMTP client/server relationship

Some of the more popular standards are:

- A standard for the exchange of mail between two computers (STD 10/RFC 821), which specifies the protocol used to send mail between TCP/IP hosts. STD 10/RFC 821 dictates that data sent via SMTP is 7-bit ASCII data, with the high-order bit cleared to zero. This standard is SMTP itself.
- A standard (STD 11) on the format of the mail messages, contained in two RFCs. RFC 822 describes the syntax of mail header fields and defines a set of header fields and their interpretation. RFC 1049 describes how a set of document types other than plain text ASCII can be used in the mail body (the documents themselves are 7-bit ASCII containing imbedded formatting information (PostScript, Scribe, SGML, TEX, TROFF, and DVI are all listed in the standard). The official protocol name for this standard is MAIL.
- A standard for the routing of mail using the Domain Name System, described in RFC 974. The official protocol name for this standard is DNS-MX.
- Multipurpose Internet Mail Extensions (MIME), defined in RFCs 2045 to 2049, which specifies a mechanism for encoding text and binary data as 7-bit ASCII within the mail envelope defined by RFC 822.
- SMTP service extensions, which define a mechanism to extend the capabilities of SMTP beyond the limitations imposed by RFC 821.

The list of mail RFC standards is much longer, but is out of scope for this book. For complete details on the standards and how they relate to each other, refer to *TCP/IP Tutorial and Technical Overview*, GG24-3376.

7.1.3 How can z/OS mail services be applied?

There are three scenarios that can be applied to the z/OS SMTP server:

- It can be set up to distribute mail on an IP network.
- It can also perform the role of a gateway server transferring mail between the IP network and a local NJE network.
- In addition to the IP network and an NJE gateway server, it can also be configured to send unresolved non-local mail or all non-local mail to a Mail Transfer Agent (MTA), sometimes called a relay server.

7.2 z/OS SMTP as a mail server

The z/OS SMTP mail server topics in this section are:

- Description of z/OS SMTP server
- Configuration tasks for the z/OS SMTP server
- Verification of the z/OS SMTP server

7.2.1 Description of z/OS SMTP server

z/OS SMTP provides a method of originating and receiving electronic mail through the TCP/IP network using the mainframe.

Automation packages may take advantage of the SMTP functions to send alert messages to systems personnel and application administrators when batch jobs fail or critical situations arise that could adversely affect the integrity of the system environment.

A mainframe JES/NJE system that does not have a TCP/IP-capable SMTP client or server running on it can still participate in sending and receiving mail by using an SMTP gateway server on another system image.

As long as one system within a sysplex of mainframes has an SMTP server running, all sysplex members that share the sysplex's JES spool can participate in the electronic mail delivery functions.

The section includes the following topics:

- Description of the z/OS SMTP server environments
- Dependencies of the z/OS SMTP server
- Considerations for using the z/OS SMTP server

Description of the z/OS SMTP server environments

The mail distribution environments of the z/OS SMTP server are depicted in Figure 7-3.

Archived

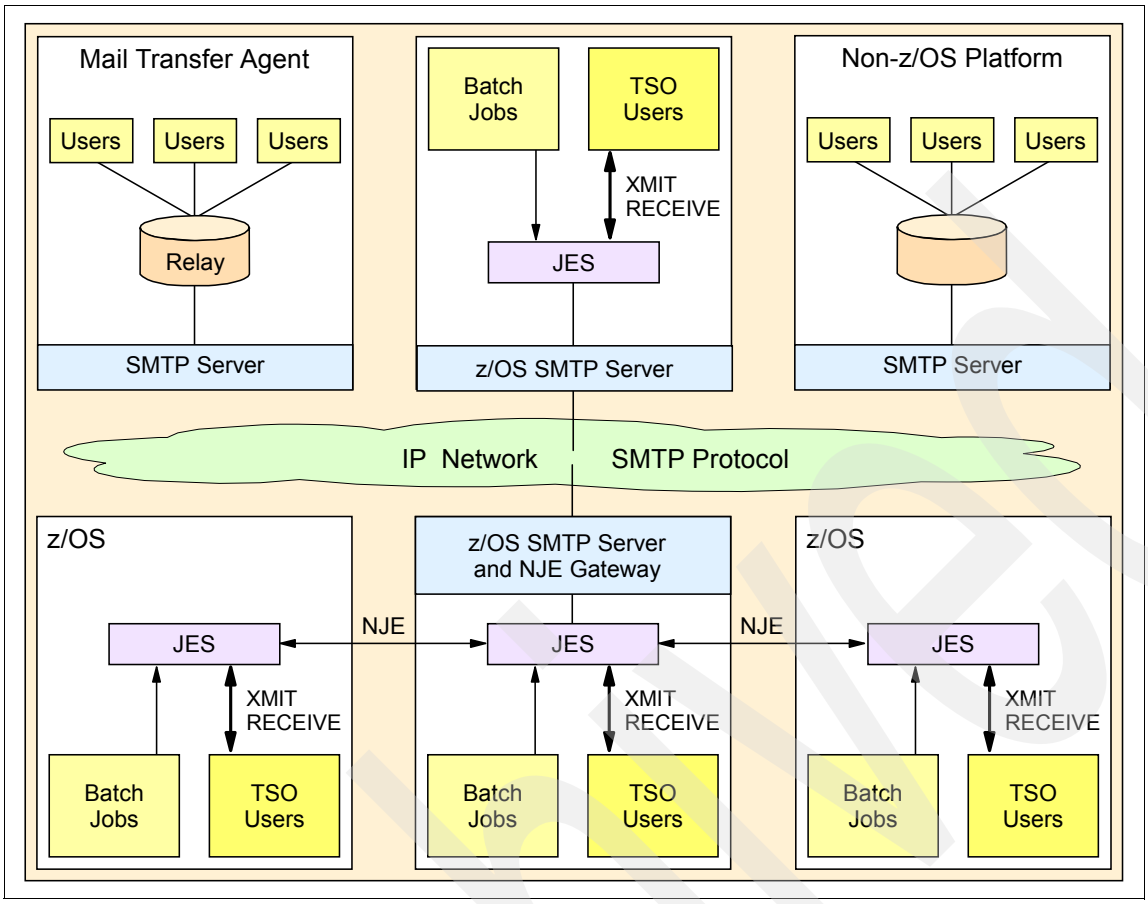


Figure 7-3 SMTP mail server: IP, NJE, relay relationships

IP network

SMTP (that is, STD 11/RFC 821) is based on end-to-end delivery. An SMTP client contacts the destination host's SMTP server directly to deliver the mail. It keeps the mail item being transmitted until it has been successfully copied to the recipient's SMTP. This is different from the store-and-forward principle that is common in many mailing systems, where the mail item may pass through a number of intermediate hosts in the same network on its way to the destination and where successful transmission from the sender only indicates that the mail item has reached the first intermediate hop.

NJE network

This setup enables NJE users to send and receive mail to and from users on TCP networks. In various implementations, there is a possibility to exchange mail between the TCP/IP SMTP mailing system and the local JES NJE spools. This SMTP server function is a mail *gateway* or mail bridge. Sending mail through a mail gateway can alter the end-to-end delivery specification, because SMTP will only guarantee delivery to the mail-gateway host, not to the real destination host, which is located beyond the TCP/IP network. When a mail gateway is used, the SMTP end-to-end transmission is host-to-gateway, gateway-to-host, or gateway-to-gateway. The behavior beyond the gateway is not defined by SMTP.

Relay server usage

When a client originates outbound mail destined for an unknown domain (unknown to the system where the SMTP server is running), the SMTP server can optionally forward the mail to a relay server that

will be able to use special SMTP DNS lookup records to determine how and where to forward the otherwise undeliverable mail. In addition, the SMTP server can be configured to forward all *out of domain* outbound mail to a relay server.

Dependencies of the z/OS SMTP server

The SMTP server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E is issued and the server terminates.

SMTP requires access to the JES spool. It uses JES utilities to create, read, write, and purge data sets from the JES spool. JES exit programs might interfere with SMTP functions.

The system security server, such as RACF, must have the SMTP started task name defined and authorized for use with the JES spool.

JES parameters must be set up in a way that mail can be sent to SMTP and that local mail can be placed on the JES spool for local users.

SMTP can be a large user of DASD. Multiple files are created for each mail note processed. Even though these files are temporary and are deleted once a note is delivered, DASD volume management is necessary to avoid contention for resources with other applications running on the system.

DASD management packages should be run only when SMTP is down.

SMTP requires special translation tables. The ASCII LineFeed character (X'0A') needs to be translated to an EBCDIC LineFeed (X'25'). Make sure that the proper translate table is available to the SMTP server address space.

The use of a relay server requires a connection over the IP network, having permission to access the relay server and to send forwarded mail to it. There must be special DNS records, called Mail Exchange (MX) records, on the DNS servers in order for the SMTP server to determine the next hop, or the next relay server in the forwarding path.

Considerations for using the z/OS SMTP server

If you have specified PROFILE NOINTERCOM in your TSO user ID's profile you do not receive some SMTP server messages.

The TSO interactive interface of SMTP is command line prompt driven, and you must know the format of the subcommands and specific syntax to be followed for the data content.

The batch interface assumes a data set has been created, containing SMTP mail commands, and submitted to the JES spool where SMTP reads the commands and processes them without interactive input or control from the originating user.

Once mail is delivered to a relay server, that mail is consider delivered by the SMTP server that forwarded the mail. It then becomes the responsibility of the relay server.If the relay server cannot be contacted, then the SMTP server must have a method and a procedure for storing and managing undeliverable mail. The sender must be notified, and attempts to retry delivery must be made until successful delivery is achieved, or until retry attempts are exhausted.

7.2.2 Configuration tasks for the z/OS SMTP server

There are common steps to set up the SMTP server regardless of how it is to be used or what role it plays. There are some optional steps to perform when the server is to perform special roles like that of an NJE gateway server, or when it will be forwarding mail to a relay server. The tasks to configure the z/OS SMTP server are:

- Update the TCPIP profile configuration data set
- Update RACF to define the SMTP started task
- Customize the SMTP server procedure JCL
- Create the SMTP server configuration data set
- Customize the SMTPNOTE CLIST for TSO support
- Customize the SYS1.PARMLIB(IKJTSOxx) member
- Determine whether NJE Gateway support is necessary: (NJE)
- Enable SMTP domain name resolution
- Define a relay server if one is planned
- Create an SMTP user exit to define and filter spam mail, if necessary

Update the TCPIP profile configuration data set

The AUTOLOG and PORT statements should be updated to indicate the action and support that the stack needs to provide for the SMTP server. AUTOLOG indicates whether the stack should initially start the SMTP started task. PORT provides a port reservation for the port number that the SMTP server listens on. The default is port 25. If your SMTP server is to listen on a different port (other than port 25), specify the port number on the PORT statement in the server's configuration file.

Update RACF to define the SMTP started task

Every started task must be assigned a user ID, and that user ID must be granted authority to access the required resources used by the started task. This discussion assumes RACF is the security subsystem being used. If another security product is used, refer to its manuals for equivalent setup instructions. Before SMTP can be started, security for the procedure name and its associated user ID must be defined. Review the sample file SEZAINST(EZARACF) that contains sample security statements for this effort.

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

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

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

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

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

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

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

1. Add the user to RACF:

   ```
   ADDUSER SMTP
   ```

2. Permit the user ID:

   a. Create a BPX.SUPERUSER FACILITY class profile, if it does not already exist:

      ```
      RDEFINE FACILITY BPX.SUPERUSER
      ```

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

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

   c. Permit the user to the class:

      ```
      ALTUSER SMTP OMVS(UID(25) PROGRAM ('/bin/sh') HOME('/'))
      PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SMTP) ACCESS(READ)
      ```

      In this example, the user ID is SMTP and the UID is 25. The UID can be any nonzero number. UID 25 was used to match the well-known SMTP port number.

   d. Refresh the FACILITY class:

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

Customize the SMTP server procedure JCL

A sample of the procedure is in hlq.SEZAINST(SMTPPROC). Customize data set names to meet standards of your installation. There are some follow-on steps below that may require you to add one or more DD statements to this procedure. Store your updated procedure in a system procedure library and make certain its name matches the name you put on the AUTOLOG and PORT statements in the TCPIP profile configuration data set.

Our SMTP server procedure is shown in Figure 7-4.

```
//SMTPB PROC MODULE=SMTP,DEBUG=,PARMS='NOSPIE/',SYSERR=SYSERR
//*
//* Turn on SMSG support
//*
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SMTPB EXEC PGM=MVPMAIN,
//            PARM='&MODULE,PARM=&DEBUG,ERRFILE(&SYSERR),&PARMS',
//            REGION=0M,TIME=1440
//*STEPLIB  DD DSN=SYS1.SEZATCP,DISP=SHR
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//LOGFILE  DD SYSOUT=*
//SMTPNJE  DD DSN=TCPIP.SMTPNJE.HOSTINFO,DISP=SHR
//CONFIG   DD DSN=TCPIP.TCPPARMS(SMTPB&SYSCLONE.),DISP=SHR
//*SECTABLE DD DSN=TCPIP.SMTP.SECTABLE,DISP=SHR
//*SMTPRULE DD DSN=TCPIP.SMTP.RULE,DISP=SHR
//SYSTCPD  DD DSN=TCPIP.TCPPARMS(DATAB&SYSCLONE.),DISP=SHR
```

Figure 7-4 SMTP server Proc JCL

Create the SMTP server configuration data set

A sample SMTP configuration member is in hlq.SEZAINST(SMTPCONF). Use this to set up a standard SMTP server. Once you are comfortable with the basic definitions, you can add optional functions such as the NJE gateway and relay server statements. For statement syntax, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For a discussion of statement use, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Figure 7-5 shows unique parameters for this server instance.

Note: Define a RACF valid user id to *BadSpoolField* parameter is required for SMTP customization. But with the design of SMTP, use the SMTP start procedure owner's id will get this message.

```
EZA5165E Invalid BadSpoolFileId Userid: SMTP
```

When SMTP detects a bad spool file, it will put it back on the JES spool with a destination of the userid defined in badspoolfield. If this userid is itself, this will cause a mail loop, because SMTP will take it back off the queue (because it is destined to itself) and keep re-processing it. To prevent this error, in the SMTP source code and there is a specific test to see if badspoolfield is itself. If it is, it issue the EZA5165E message. So the bottom line is badspoolfield can be any valid userid but the owner's id of its start procedure.

Archived

```
;*********************************************************************
;  Defaults that normally aren't changed:
;*********************************************************************
PORT 25
BADSPOOLFILEID CS07
LOG
INACTIVE 180
FINISHOPEN 120
RETRYAGE 3
WARNINGAGE 1
RETRYINT 20
MAXMAILBYTES 524288
RESOLVERRETRYINT 20
RCPTRESPONSEDELAY 60
TEMPERRORRETRIES 0
SPOOLPOLLINTERVAL 30
TIMEZONE SYSTZ
;*********************************************************************
;  INSTALLATION SPECIFIC STATEMENTS
;*********************************************************************
NJENODENAME WTSCPLX5
ALTTCPHOSTNAME SC31MAIL
MAILFILEDSPREFIX SMTPB
MAILFILEUNIT SYSDA
POSTMASTER CS07
SMSGAUTHLIST
    SMTP
    CS07
    CS01
ENDSMSGAUTHLIST
;*********************************************************************
; CONFIGURATION FOR A TYPICAL NJE TO TCP/IP MAIL GATEWAY.
;*********************************************************************
GATEWAY                ; ACCEPT MAIL FROM AND DELIVER MAIL TO NJE HOSTS
NJEDOMAIN SCNJENET     ; ANY TWO NAMES WE WANT.  OTHERS MUST USE THESE
ALTNJEDOMAIN SCNJENET  ; USER02%SCNJENET@SC31MAIL.RALEIGH.IBM.COM
NJEFORMAT PUNCH        ; NJE RECIPIENTS RECEIVE MAIL IN PUNCH FORMAT
NJECLASS A             ; SPOOL CLASS FOR MAIL DELIVERED BY SMTP TO THE
LOCALFORMAT NETDATA    ; LOCAL RECIPIENTS GET MAIL IN NETDATA FORMAT
;                      ; NETDATA ALLOWS TSO RECEIVE TO BE USED WITH MAIL
LOCALCLASS A           ; SPOOL CLASS FOR LOCAL MAIL DELIVERED BY SMTP
REWRITE822HEADER YES NOPRINT
;*********************************************************************
; The RESTRICT statement specifies addresses of users who cannot
; utilize SMTP services.
;*********************************************************************
RESTRICT RETURN          ; Return mail from restricted users
  cs09@us.ibm.com ; DON'T ACCEPT ANY MAIL FROM PRINCE CHARMING
  cs09@SCNJENET          ; VIA NJE OR TCP NETWORK.
  cs09@*                ;THIS LINE TAKES THE PLACE OF PREVIOUS 2 LINES
 *@badsite              ;DON'T ACCEPT MAIL FROM ANYONE AT HOST CASTLE
ENDRESTRICT
;*********************************************************************
;      SEND ALL NON-LOCAL MAIL TO AN MTA RELAY SERVER
;*********************************************************************
IPMAILERADDRESS 10.12.4.221
RESOLVERUSAGE YES
```

Figure 7-5 SMTP server configuration data set: unique parameters

Customize the SMTPNOTE CLIST for TSO support

If you plan to support TSO users in creating outbound mail, set up the SMTPNOTE CLIST. There is a sample in hlq.SEZAINST(SMTPNOTE). It is a command-line interface that prompts for required information to allow a TSO user to set up a note and have it delivered to its destination. The clist collects the necessary information from the user and builds a file that is then written to the JES spool for the SMTP server. The server discovers the note on the spool and processes the SMTP command within the file.

Users can also use IEBGENER in batch to generate a file that contains all the same SMTP commands necessary to send a note through SMTP to the IP or NJE network.

Whether the SMTPNOTE clist will be used, or the batch method, or both, values for the following items must be established and then assigned:

DDNAME This is the temporary data set used to build the SMTP commands for the note. SMTPNOTE refers to this data set as the INPUT data set. Once the commands are built and the user indicates the note is ready for delivery, SMTPNOTE transmits the contents of this file (which are SMTP commands) to the JES spool destined for the SMTP server.

TEMPDSN This is the name of the data set allocated for the DDNAME above. Make certain the data set name ends with the low-level qualifier of .TEXT and do not fully qualify the name. By not fully qualifying the name, the clist prefixes the name with the user's TSO *user ID*.

HOSTNAME This is usually the NJE node name of the system on which the SMTPNOTE clist runs.

SMTPNODE This is the NJE node name of the system where the SMTP server runs. If the SMTP server is acting as an NJE gateway server, then it may be on a different NJE node than where the SMTPNOTE clist is running. SMTPNODE must always point to the node of the SMTP server.

SMTPJOB This is the started task job name of the SMTP server itself. The SMTPNOTE clist uses TSO XMIT to send the note to the SMTP server. XMIT uses JES facilities and the JES spool. The started task name must not be the same as any node name defined to JES. It cannot begin with the characters R, RM, or RMT because JES could get confused and think that the file should be delivered to a JES Remote instead of the SMTP server. The XMIT command sends the note file to the JES spool using the two values you specify for "*SMTPNODE.SMTPJOB*". That spool location must not be processed by any other service than the SMTP address space.

TIMEZONE This is for the system on which this SMTPNOTE clist runs. The value of SYSTZ will allow the code to dynamically retrieve the value of the timezone from the system Communication Vector Table (CVT) control block.

ATSIGN This specifies a single-byte representation of the at symbol (@) for foreign languages.

For use of the SMTPNOTE clist, see *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780.

Customize VMCF and TNF

The VMCF and TNF is required for starting SMTP, for detail information about VMCF and TNF customization, refer to *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532

Archived

Customize the SYS1.PARMLIB(IKJTSOxx) member

The SMTPNOTE clist uses the TSO XMIT command. The XMIT command uses JES facilities to accomplish the transfer to spool. The TRANSREC statement must contain the correct node name. As an alternative, the NODESMF parameter can be coded as NODESMF((*,*)). *,* specifies that the node name is to be retrieved dynamically from JES. This specification is recommended because it eliminates the need to specify static values for node name and smfid. For more information about the TRANSREC statement, refer to *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

Determine whether NJE Gateway support is necessary: (NJE)

If you transmit messages to the JESSPOOL destined for the SMTP server using TSO XMIT or using IEBGENER, then you need to implement the SMTP server as a local NJE gateway. In order to implement the SMTP server as an NJE Gateway, some terminology needs to be discussed. Refer to the diagram in Figure 7-6 for a list of the NJE Gateway related statements.

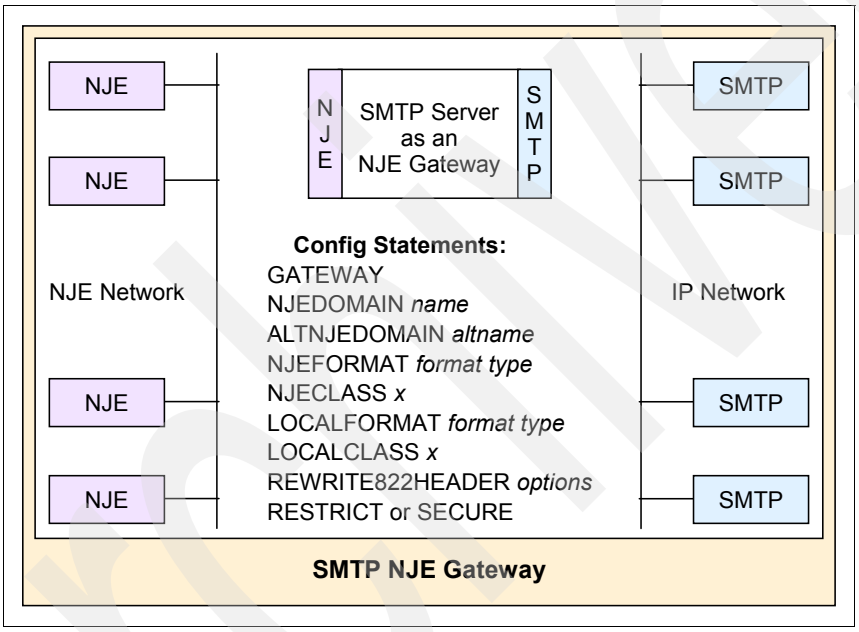


Figure 7-6 SMTP server as an NJE gateway server

If you intend to use the SMTP server as a gateway to the NJE network, these tasks must be considered:

- Customize the SMTP mail headers, if necessary: (NJE).

  Usually, the default header rules supplied by IBM are sufficient for most NJE traffic. However, if you have a special condition that is not covered by the default rules, see the comprehensive discussion of customizing mail headers in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

- Set up the TCP-to-NJE gateway function: (NJE).

  Assign values for and plan to use each of the following statements in the SMTP configuration data set:

  – GATEWAY indicates that this SMTP server is an NJE gateway.
  – NJENODENAME is the node name of the local JES NJE system.
  – NJEDOMAIN sets the domain name of the NJE network to what you want it to be.
  – ALTNJEDOMAIN is an alternate domain name of the NJE network (synonym).

- NJECLASS is the JES spool data set class for mail delivered on the NJE network.
- NJEFORMAT is the JES spool data set format to be used.

► Plan to use the NJENODENAME statement within the SMTP configuration file.

Do not plan to depend on the setting of the VMCF node name value either in SYS1.PARMLIB(IEFSSNxx) or set by the SYS1.PROCLIB(EZAZSSI) procedure. These values for node name or system name could be and usually are different from the NJE node name. SMTP requires you to know the actual NJE node name value. Do it the easy way by specifying it with the NJENODENAME statement.

Note: The NJENODENAME statement, if specified, must precede any of the following statements in the SMTP Configuration Data Set:

► ALTNJEDOMAIN
► MAILER
► NJEDOMAIN
► SMSGAUTHLIST

► Create the required NJE host table data set named *hlq*.SMTPNJE.HOSTINFO.

SMTP cannot accept mail destined to a node name that JES does not have defined. The SMTP gateway server requires this data set to determine which NJE nodes are defined to JES and are thus permitted to participate in mail transfers. In order to build this data set you must run the following TSO command and point it to the JES initialization data set member that contains the JES node definitions:

```
TSO SMTPNJE 'SYS1.PARMLIB(JES2PARM)'
```

The command scans the member for node definitions in HASPPARM DD statement of JES2 start procedure and builds the required file in *userid*.SMPTNJE.HOSTINFO, you can change to name to your consistent HLQ.

For JES3; add the parameter at the end of the command to point to JES3, as the following shows:

```
TSO SMTPNJE data.set.pds.name(member) (JES3
```

The default is JES2.

► Add the required //SMTPNJE DD statement to the SMTP server procedure JCL, pointing it to the HOSTINFO data set just created:

```
//SMTPNJE DD DSN=hlq.SMTPNJE.HOSTINFO,DISP=SHR
```

► Do you want to define a SECURE NJE Gateway?

A SECURE statement can be added to the SMTP server configuration data set to define the server as a secure gateway between the TCP/IP network and the NJE network. When the server is operating in secure mode, only those NJE addresses in the SMTP security table are allowed to use the mail services of this server. The SMTP server rejects mail to or from an unauthorized user. An unauthorized user is one whose user ID is *not* in the table. This table is coded as records in the required data set:

```
mailfiledsprefix.SMTP.SECTABLE with LRECL=255 and RECFM=VB
```

And it is pointed to by adding the required DD statement to the SMTP server proc:

```
//SECTABLE DD DSN=mailfiledsprefix.SMTP.SECTABLE,DISP=SHR.
```

The *mailfiledsprefix* hlq is also defined within the SMTP configuration data set.

You must create a second required data set. It is used when NJE mail is rejected. Its contents are used as a memo note that is sent to the unauthorized user whose mail is rejected. The name of this data set is:

mailfiledsprefix.SECURITY.MEMO with LRECL=255 and RECFM=VB

It is dynamically allocated by the SMTP server when needed. You do *not* add a DD statement to the SMTP server procedure JCL.

For examples and details on the format and syntax of the records for both data sets, you must refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

- Do you want to define a RESTRICT NJE Gateway?

  A RESTRICT statement can be added to the SMTP server configuration data set to indicate those user IDs who are not allowed to use the mail services of this server. You code a list of user IDs within the RESTRICT list statement. For details on the syntax of the RESTRICT statement you must refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Note: RESTRICT and SECURE are both optional settings. However, they are mutually exclusive.

Figure 7-7 shows NJE-related parameters.

```
NJENODENAME WTSCPLX5
...
SMSGAUTHLIST
    SMTP
    CS07
    CS01
ENDSMSGAUTHLIST
...
;*********************************************************************
; CONFIGURATION FOR A TYPICAL NJE TO TCP/IP MAIL GATEWAY.
;*********************************************************************
GATEWAY                ; ACCEPT MAIL FROM AND DELIVER MAIL TO NJE HOSTS
NJEDOMAIN SCNJENET     ; ANY TWO NAMES WE WANT.  OTHERS MUST USE THESE
ALTNJEDOMAIN SCNJENET  ; USER02%SCNJENET@SC31MAIL.RALEIGH.IBM.COM
NJEFORMAT PUNCH        ; NJE RECIPIENTS RECEIVE MAIL IN PUNCH FORMAT
NJECLASS A             ; SPOOL CLASS FOR MAIL DELIVERED BY SMTP TO THE
LOCALFORMAT NETDATA    ; LOCAL RECIPIENTS GET MAIL IN NETDATA FORMAT
;                      ; NETDATA ALLOWS TSO RECEIVE TO BE USED WITH MAIL
LOCALCLASS A           ; SPOOL CLASS FOR LOCAL MAIL DELIVERED BY SMTP
REWRITE822HEADER YES NOPRINT
```

Figure 7-7 SMTP server configuration data set: NJE parameters

Figure 7-8 shows restrict and secure related settings.

```
;*********************************************************************
; Use the SECURE statement if this SMTP machine is to run as an SMTP-
; to-NJE Secure Gateway.  Only users in the SMTP.SECTABLE data set
; will be allowed to send mail, all other mail will be returned or
; rejected.  Note that the contents of dataset
; mailfiledsprefix.SECURITY.MEMO will be sent to NJE users that are
; not authorized to use the gateway.
;*********************************************************************
; SECURE
;*********************************************************************
; The RESTRICT statement specifies addresses of users who cannot
; utilize SMTP services.
;*********************************************************************
RESTRICT RETURN          ; Return mail from restricted users
  cs09@us.ibm.com        ; DON'T ACCEPT ANY MAIL FROM PRINCE CHARMING
  cs09@SCNJENET          ; VIA NJE OR TCP NETWORK.
  cs09@*                ;THIS LINE TAKES THE PLACE OF PREVIOUS 2 LINES
 *@badsite              ;DON'T ACCEPT MAIL FROM ANYONE AT HOST CASTLE
ENDRESTRICT
```

Figure 7-8 SMTP server configuration data set: secure and restrict parameters

Enable SMTP domain name resolution

The SMTP server configuration statement RESOLVERUSAGE indicates whether domain name resolution is to be performed. If name resolution is desired, make certain that the //SYSTCPD DD statement is in the SMTP server procedure JCL and that it points to a valid TCPDATA file containing correct DNS server information. If name resolution is not desired, you must code RESOLVERUSAGE NO. For a complete discussion about how DNS services are used by the SMTP server and how it processes DNS MX type records, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Define a relay server if one is planned

Non-local mail must go through a Mail Transfer Agent (MTA) to get to another host. SMTP supports the following configuration statements to assist in forwarding non-local mail:

- IPMAILERNAME, for non-local mail destined for SMTP servers in the IP network using a host name
- IPMAILERADDRESS, for non-local mail destined for SMTP servers in the IP network using a static IP address
- MAILER, for non-local mail destined for SMTP servers in the NJE network using the JES spool

There is a special situation where you may want to send *all* non-local mail to a relay server, and not just *unresolved* non-local mail. For a detailed discussion about how to direct all non-local mail to a relay server, see the topic on sending non-local messages to other mail servers in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Figure 7-9 shows relay server related parameters.

```
;*********************************************************************
;      SEND ALL NON-LOCAL MAIL TO AN MTA RELAY SERVER
;*********************************************************************
IPMAILERADDRESS 10.12.4.221
RESOLVERUSAGE YES
```

Figure 7-9 SMTP server configuration data set: relay server parameters

Create an SMTP user exit to define and filter spam mail, if necessary

There is a sample user exit in hlq.SEZAINST(SMTPEXIT). For details on implementing and managing the exit, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

7.2.3 Verification of the z/OS SMTP server

To receive a detailed trace on how SMTP is resolving a particular host name, you can issue the TSO SMSG SMTP TRACE command in TSO, or the MVS MODIFY (F) command, or use a SYSTCPT DD statement in the SMTP cataloged procedure. Samples of these methods are:

```
TSO SMSG SMTPB TRACE
F SMTPB,SMSG,TRACE
//SYSTCPT DD SYSOUT=*
```

Note: The TSO SMSG command works when issued from TSO only and should not be issued from the operator console. SMSG SMTP is not supported in batch. It uses VMCF and the PASCAL interface to queue information and will not print the information to a DD card. Issue the MVS MODIFY (F) command from the MVS console or an application console interface such as that provided within SDSF or Netview.

An automation tool could issue a MODIFY command and then notify an administrator when a large amount of mail is queued for SMTP, based on the output of the command when using the NUMQueue parameter.

You can also add the TRACE RESOLVER statement when configuring the TCPIP.DATA data set, but this will trace name resolution for all the applications using the name server. To prevent the console log from becoming too large, only use the TRACE RESOLVER statement for debugging.

TIMEZONE parameter

To verify that the value of SYSTZ set to the TIMEZONE parameter in a SMTP server configuration works correctly, we submitted a batch SMTP. Example 7-1 shows messages in a spool file created after we submitted a batch job. We can see -0400 (1) instead of the printable zone name such as EST. (The 0400 represents HH:MM.)

Example 7-1 SMTP Spool file

```
Received: from WTSC31B.ITSO.IBM.COM by WTSC31B.ITSO.IBM.COM (IBM MVS SMTP CS)
   with BSMTP id JOB03510; Mon, 24 Sep 07 15:02:34 -0400 1
Date: Mon, 24 Sep 07 15:02:34 -0400 1
From: <CS10@WTSC31B.ITSO.IBM.COM>

THIS IS A TEST MAIL.
```

7.3 Sendmail and popper as mail servers

In this section we establish sendmail and popper as mail servers. The following topics are discussed:

- Description of sendmail and popper
- Configuration tasks for sendmail and popper
- Verification of sendmail and popper setup

7.3.1 Description of sendmail and popper

Sendmail is an industry standard mail application that is widely used on the Internet. The sendmail application included with the z/OS Communications Server is based on sendmail Version 8.12.1. Popper is a Post Office Protocol 3 (POP3) mail delivery agent. It allows a remote user to use a remote mail user agent (MUA) to read, compose, and manage e-mail that has been delivered to z/OS. Refer to Figure 7-3 on page 201 to review the role of a mail server.

Sendmail and popper are full-featured industry standard mail applications. Sendmail is also capable of acting as a client that can be used to send e-mail. Sendmail supports MIME attachments and also has built-in security features such as TLS/SSL sockets.

Note: Due to the complexity of sendmail, we highly recommend that you become familiar with the industry-accepted publication about sendmail: *sendmail, 3rd Edition* , from O'Reilly & Associates, Inc.

The following topics are discussed in this section:

- Dependencies for sendmail and popper
- Considerations when using sendmail and popper
- MTA, MUA, and MDA
- M4 preprocessor
- Alias file
- The statistics file
- The help file
- The queue directory
- The sendmail configuration file
- Popper

Dependencies for sendmail and popper

Sendmail needs information that, for example, defines the local mailer program, defines which file system is responsible for deferred delivery, and defines which file contains information about alias names and real names. The only required file is the sendmail configuration file (usually /etc/sendmail.cf). However, that file may list other directories and files that must be created before sendmail can start. In order to start sendmail using the example configuration file, you must create a configuration file, a mail queue directory, the /etc/mail directory, and a local-host-names file.

Considerations when using sendmail and popper

Sendmail can be very complex. Sendmail is incapable of interfacing with JES. If you want to submit batch job e-mail, then you should use SMTP rather than sendmail.

Note: Sendmail is hard-coded to use ISO8859-1 and IBM-1047 encoding. It does not use DBCS character sets at all. If you want to use DBCS character sets, use MIME.

MTA, MUA, and MDA

When discussing sendmail and popper, other books refer to Mail Transfer Agents (MTA), Mail User Agents (MUA), and Mail Delivery Agents (MDA). Let us take a moment to define these acronyms:

- Mail Transfer Agent (MTA)

  This is any application that sends a prepared e-mail message to a remote MTA. Sendmail, Microsoft Exchange, and Lotus® Domino® are examples of MTAs.

- Mail User Agent (MUA)

  This is any application that allows a user to read, compose, and manage e-mail. Lotus Notes®, Microsoft Outlook®, and Netscape Navigator are all applications that have MUA capabilities.

- Mail Delivery Agent (MDA)

  This is any application that sits between an MUA and MTA and manages delivering a message from a MTA to an MUA. A MDA is an optional piece of an e-mail system. Popper is an example of an MDA.

In z/OS UNIX, sendmail is a MTA, **`/bin/mail`** is a command-line MUA, and popper is an MDA. In our example configuration, we use sendmail as a MTA, Microsoft Outlook Express as an MUA, and popper as an MDA.

M4 preprocessor

The m4 macro processor is a front-end processor for many programming languages. Besides replacing one string of text with another, the m4 macro processor provides the following features:

- Arithmetic capabilities
- File manipulation
- Conditional macro expansion
- String and substring functions

The m4 macro preprocessor can be given input that will generate a z/OS UNIX sendmail configuration file. It takes as input a user-defined master configuration source file (.mc file) defines mail delivery mechanisms using files provided in the samples directory. When you run the m4 preprocessor, you need files that contain input definitions and an output file that will be your sendmail configuration file. The input files are:

/m4/cf.m4 Provides support for different include files such as cfhead.m4 and proto.m4.

/cf/sample.mc References other files and their locations. The sample.mc file is located in the directory /usr/lpp/tcpip/samples/sendmail/cf.

You may use the cf.m4 and sample.mc files to create your own sendmail.cf output file or you may use the pre-generated sample configuration file. In our environment, we first used the sample configuration file to get sendmail started, and then later used m4 to generate a new configuration file.

Alias file

The alias file is used to convert an alias name to another recipient's name. A large number of names could potentially be listed in an alias file. In order to efficiently deal with a potentially large alias file, sendmail uses an alias database file created from a alias file. The database version of the alias file significantly improves lookup speed. A database file is created from the alias file with the **`/usr/sbin/newaliases`** or **`/usr/sbin/sendmail -bi`** commands.

The statistics file

The sendmail.st statistics file is used by sendmail to record the number and sizes of all incoming and outgoing mail messages handled by each delivery agent. Statistics are kept for each of the following delivery agents:

- Local delivery agent
- SMTP delivery agent
- UUCP delivery agent

Statistical information is collected if option (O) is defined in the sendmail.cf file. Sendmail does not create the statistics file; you must manually create the file first, before using the statistics option. You may view the statistics file using the commands **`mailstats -C </etc/sendmail.cf>`** or **`mailstats -s </etc/sendmail.st>`**.

Use of the statistics file is optional.

The help file

The sendmail.hf file is the help file implemented for SMTP (and ESMTP). You will find this file in the /usr/lpp/tcpip/lib directory. If you want to view this file, you can issue **`obrowse /usr/lpp/tcpip/lib/sendmail.hf`**.

Use of the help file is optional.

The queue directory

Sendmail creates a queue directory the first time sendmail is run. The location of the queue directory is specified in the sendmail configuration file. If a mail message cannot be transmitted sendmail stores it in the queue directory until the message can be transmitted successfully. Possible reasons for queuing a message include:

- The remote machine is down.
- There are temporary disk problems.
- Sendmail or another MTA on the other machine is not started.
- There are TCP communication problems.

If you have the permission to look at the queue directory, you may find it empty, which implies that all messages have been sent. Or you may find some dfxxxxxxxx and qfxxxxxxxx files, which are messages that are waiting to be sent. When a message is queued, it is split into two parts. Each part is saved in a separate file. Header information is contained in qf files. The actual body of the message in included in the df files. The **`obrowse`** command may be used to read these files. Superuser permission is normally needed.

The following is a description of the header lines:

1. Version of the qf file: V2 means above the V8.8 sendmail version.
2. Time created in seconds to limit the message to remain in the queue.
3. Determines the time to wait before retrying delivery.
4. Number of attempts for each delivery.
5. Priority when processed from the queue.
6. After a system crash the message is stored in lost+found under the referenced number in case the qf file lost its directory entry.
7. Reason why the message was stored in the queue (= deferred).
8. Full canonical name of the sender's machine.
9. Sender's address.

10. For security reasons, this is the real recipient of the message.
11. Recipient's address.
12. Header information for the return path in case the message cannot be delivered.
13. HReceived: where the message came from.
14. Header information when the message was received by the MTA.
15. Header information about the sender.
16. Header information about the full name of the sender.

All header information is based on entries in sendmail's configuration file /etc/sendmail.cf. You can also get queue information for all messages in the queue by running the sendmail command with the -bp command-line switch (printing the queue).

The sendmail program offers two different methods for processing the queue:

- Process the queue periodically with the -q and -h command-line switches.
- Process the queue once with only the -q command-line switch and then exit.

The sendmail configuration file

The sendmail configuration file is read and parsed by the sendmail program every time sendmail starts. It lists the locations of important files and specifies the default parameter values to be used within the sendmail program. The parameters within the configuration file define sendmail's behavior and contain rules and rule sets for tasks such as rewriting the mailing address. The configuration file uses a basic syntax, which consists of a command followed by a value.

Examples of configuration commands are:

M Define a mail delivery agent.
R Define rewriting rules.
H Define a header.
P Define delivery priorities.
T Define a trusted user.
D Define a macro.
O Define an option.
S Declare a rule-set start.

There are many more definitions in the sendmail.cf file, so this file can be very complex. If you browse through the configuration file found in /usr/lpp/tcpip/samples/sendmail/cf/sample.cf you will find very a complex example. In order to make it easier to start sendmail, you only need to copy the /usr/lpp/tcpip/samples/sendmail/cf/sample.cf file to /etc/sendmail.cf. The sample sendmail file was used in our tests.

The sample.cf file we used was created automatically by running the m4 macro preprocessor with the master input file in /usr/lpp/tcpip/samples/sendmail/cf/sample.mc. If you browse through this sample.mc file, you will notice that the most common information is already defined, which provides a base to start with if you later want to add to the sample sendmail.cf file.

Popper

The only configuration required for popper is to update /etc/services and /etc/inetd.conf. Using popper is discussed in detail in 7.3.3, “Verification of sendmail and popper setup” on page 221.

7.3.2 Configuration tasks for sendmail and popper

In this section we use the samples provided with the z/OS Communications Server to start sendmail. We then start popper and use a third-party MUA to retrieve e-mail from z/OS. The following tasks must be performed to start sendmail.

Important: The following tasks must be performed by the superuser ID, UID=0. If you use another user ID, then sendmail may issue the following messages:

```
EZZ9927I Permission denied (real uid not trusted).
or
dbmmap "Alias0": unsafe map file /etc/mail/aliases: EDC5111I Permission denied.
WARNING: cannot open alias database /etc/mail/aliases
```

If your ID is not a superuser ID and you forget to issue an **su** command before you create the /etc/mail directory, you can issue the command **chown 0:0 /etc/mail** to change the ownership of the /etc/mail directory and bypass these errors.

- Create the /etc/mail directory
- Create the configuration file
- Create the queue directory
- Create the local-host-names file
- Create the popper maildrop directory
- Create the aliases database
- Update /etc/inetd.conf for popper
- Update /etc/services
- Update PROFILE.TCPIP
- Create sendmail start procedure
- Start sendmail
- Verification of sendmail and popper setup

Create the /etc/mail directory

From the z/OS UNIX System Services shell, issue:

```
mkdir /etc/mail
```

Both the /etc directory and the /etc/mail subdirectory should have file permissions of 755. If they need to be changed, issue:

```
chmod 755 /etc
chmod 755 /etc/mail
```

Create the configuration file

The sample configuration file for sendmail is in /usr/lpp/tcpip/samples/sendmail/cf/sample.cf. So, to make it easier to test sendmail, we can do:

- Both with IPv4 and IPv6 supports

If your environment is already customized to support AF_INET6 in BPXPRMxx parmlib and the affinitive TCP/IP stack is also customized for IPv6, then you can simply copy the sample.cf and use this copy. From the z/OS UNIX System Services shell, issue:

```
cp /usr/lpp/tcpip/samples/sendmail/cf/sample.cf /etc/mail/sendmail.cf
```

- Only with IPv4 support

If your environment is not customized to support AF_INET6, or the TCP/IP stack is not customized for IPv6, then perform the following steps to create your own sendmail.cf:

1. Retrieve the m4 preprocessor.

   Retrieve the m4 macro preprocessor from the z/OS Toys and Tools Web page at:

   http://www.s390.ibm.com/unix/bpxa1toy.html

   Download it and FTP to the z/OS UNIX file system with binary, then place it in /tmp/m4.bin.pax.Z. Issue the following command to unpax the m4 macro preprocessor:

   ```
   pax -rzf /tmp/m4.bin.pax.Z
   ```

   Then the m4 preprocessor is created in /tmp/bin/m4 with its information file in /tmp/info/m4.info.

2. Create the .mc file.

   Copy the example .mc file from /usr/lpp/tcpip/samples/sendmail/sample.mc using the following command:

   ```
   cp /usr/lpp/tcpip/samples/sendmail/cf/sample.mc /etc/mail/sendmail.mc
   ```

   Comment out the IPv6 statement from /etc/mail/sendmail.mc file using the oedit /etc/mail/sendmail.mc command, as shown in Example 7-2.

Example 7-2 /etc/mail/sendmail.mc

```
divert(-1)
#
#  Sample configuration for z/OS
#
#
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM
# 5694-A01
# (c) Copyright IBM Corp. 1992, 2005
#
divert(0)dnl
VERSIONID(`z/OS sample configuration  2007/09/20')
OSTYPE(zOS)dnl
DOMAIN(generic)dnl
divert(-1)
#
# Set the name of the required zOS configuration file.
# A sample is shipped in /usr/lpp/tcpip/samples/sendmail/cf/zOS.cf
#
# Remove the undefine statement for confZOS_FILE to specify the
# zOS.cf file
divert(0)dnl
define(`confZOS_FILE', `/etc/mail/zOS.cf')dnl
undefine(`confZOS_FILE')dnl
divert(-1)
#
# Add WorkAroundBrokenAAA for IPv6 errors on name servers
divert(0)dnl
define(`confBIND_OPTS', `WorkAroundBrokenAAAA')dnl
divert(-1)
#
# listen for both inet and inet6 sockets
divert(0)dnl
DAEMON_OPTIONS(`Name=MTA,   Family=inet ')dnl
#DAEMON_OPTIONS(`Name=MTA-6, Family=inet6')dnl
divert(-1)
#
# define mailers
divert(0)dnl
```

```
MAILER(local)dnl
MAILER(smtp)dnl
```

3. Build the configuration file.

   To build the configuration file, go to the directory containing m4/cf.m4, and then use the m4 macro preprocessor with the following commands:

```
$ cd /usr/lpp/tcpip/samples/sendmail/cf
$ /tmp/bin/m4 ../m4/cf.m4 /tmp/sendmail.mc > /etc/mail/sendmail.cf
```

Create the queue directory

From the z/OS UNIX System Services shell, issue:

```
mkdir /usr/spool/mqueue
```

Create the local-host-names file

The local-host-names file identifies host names for which sendmail is to receive mail. We will keep this file empty for now, but it must be created in order for sendmail to start.

From the z/OS UNIX System Services shell, issue:

```
touch /etc/mail/local-host-names
```

Create the popper maildrop directory

From the z/OS UNIX System Services shell, issue:

```
mkdir /usr/mail/popper
chmod 777 /usr/mail/popper
```

Create the aliases database

The alias file maps names in e-mail addresses to local user accounts. You must list in the alias file all local users who are to receive e-mail. Our alias file is shown in Example 7-3.

Example 7-3 Contents of our /etc/mail/aliases file

```
MAILER-DAEMON:IBMUSER
postmaster:IBMUSER
cs01:CS01
cs07:CS07
nobody: /dev/null
```

You can create the file with the command **oedit /etc/mail/aliases**. Then change the file permissions with **chmod 600 /etc/mail/aliases**. Finally, run the **/usr/sbin/sendmail -bi -f /etc/mail/sendmail.cf** command to create a binary database. The message in Example 7-4 shows when you successfully create the aliases database.

Example 7-4 Messages for aliases database

```
EZZ9988I /SC31/etc/mail/aliases : 5 aliases, longest 9 bytes, 68 bytes total
```

Important: You cannot set the group write or other write bits on in the mode field of the asiases file. Otherwise, message EZZ9993I will be issued with the 265 053B006C error code for the Group writable file.

Archived

Update /etc/inetd.conf for popper

Popper is started by inetd and requires an entry in the /etc/inetd.conf file. The statement is shown in Example 7-5. If you have already started INETD as described in Chapter 6, “INETD” on page 185, then you have already completed this step.

Example 7-5 /etc/inetd.conf statement for popper

```
pop3       stream tcp nowait bpxroot     /usr/sbin/popper popper
```

Update /etc/services

/etc/services should contain the lines shown in Example 7-6. If you have already started INETD as described in Chapter 6, “INETD” on page 185, then you have already completed this step.

Example 7-6 /etc/services entries

```
smtp            25/tcp          mail
pop3           110/tcp          popper
```

If the lines are not present, then add them.

Update PROFILE.TCPIP

We need to reserve TCP port 25 for sendmail. The easiest way to accomplish this is to reserve port 25 TCP to omvs. Because popper uses port 110 TCP, and is started by INETD, we need to reserve port 110 to INETD1. Example 7-7 shows our PROFILE.TCPIP statements.

Example 7-7 PORT statements in PROFILE.TCPIP

```
25 TCP OMVS
110 TCP INETD1
```

Create sendmail start procedure

To manage the sendmail flexibly, we create a z/OS start procedure SENDMAIL with BPXBATCH utility as Example 7-8 shows:

Example 7-8 SYS1.PROCLIB(SENDMAIL)

```
//SENDMAIL PROC SMAILENV=SMAILENV
//SENDMAIL EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//   PARM='PGM /usr/sbin/sendmail -bd -q5m -C /etc/mail/sendmail.cf &'
//STDOUT   DD PATH='/tmp/sendmail-stdout',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=SIRWXU
//STDERR   DD PATH='/tmp/sendmail-stderr',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=SIRWXU
//STDENV   DD DSN=TCPIP.TCPPARMS(&SMAILENV.),DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.TCPPARMS(DATAB&SYSCLONE.)
```

Use the STDENV DD statement in the start procedure to point to a data set that specifies the environment variables for sendmail, as Example 7-9 shows:

Example 7-9 TCPIP.TCPPARMS(SMAILENV)

```
_BPX_JOBNAME=SENDMAIL
```

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIP
```

Define the RACF profile SENDMAIL.* in the STARTED class with a superuser ID as an owner in its STDATA:

```
ADDGROUP SNDMGRP OMVS(GID(26))
ADDUSER SENDMAIL DFLTGRP(SNDMGRP) NOPASSWORD OMVS(UID(0) HOME('/'))
RDEFINE STARTED SENDMAIL.* STDATA(USER(SENDMAIL))
SETROPTS RACLIST(STARTED) REFRESH
PERMIT BPX.DAEMON CLASS(FACILITY) ID(SENDMAIL) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Start inetd

If you have already started INETD as described in Chapter 6, "INETD" on page 185, then you have already completed this step. Otherwise, issue the system command **s inetd** or the following shell commands from the z/OS UNIX System:

```
export _BPX_JOBNAME=INETD
export _BPXK_SETIBMOPT_TRANSPORT=TCPIP
/usr/sbin/inetd &
```

Note: If you already have an INETD running and it does not have the popper customized, then after you add pop3 statements into its configuration files, you need to recycle the INETD. To stop the INETD, issue the following command in the z/OS UNIX System Services shell:

```
kill -TERM `cat /etc/inetd.pid`
```

Start sendmail

To start the sendmail, issue the system command **s sendmail**. The BPXBATCH utility passes the shell command in the PARM= field to OMVS with its environment variables in the STDENV DD statement. It is the same as issuing the following command in the z/OS UNIX System Services shell:

```
export _BPX_JOBNAME=SENDMAIL
export _BPXK_SETIBMOPT_TRANSPORT=TCPIP
/usr/sbin/sendmail -bd -q5m -C /etc/mail/sendmail.cf &
```

This method of starting sendmail leaves a running process that handles mail and processes any queued mail once every hour.

To stop the sendmail, issue the following command in the z/OS UNIX System Services shell:

```
kill -TERM `cat /etc/mail/sendmail.pid`
```

7.3.3 Verification of sendmail and popper setup

As shown in Example 7-10, we can use the **ps** command to verify that sendmail has started.

Example 7-10 Output of ps command

```
50462948 ttyp0000  0:00 /usr/sbin/sendmail
```

Next, as shown in Example 7-11 on page 221, we can use **netstat** to determine whether sendmail has put a listener up on port 25 TCP.

Example 7-11 Output of netstat

```
SENDMAIL 0000003E Listen
```

```
Local Socket:   0.0.0.0..25
Foreign Socket: 0.0.0.0..0
```

Finally, we use the freely available e-mail client Mozilla Thunderbird as an MUA to check our e-mail. We obtained Thunderbird 2 from http://www.mozilla.com and installed it on a PC. While installing Thunderbird, we were prompted to set up an e-mail account. The following five screen captures show the account settings needed to allow Thunderbird to access sendmail and popper running on our z/OS system.

The first panel of the configuration wizard in Mozilla Thunderbird asks what type of account is being set up. We are setting up an e-mail account, as shown in Figure 7-10.

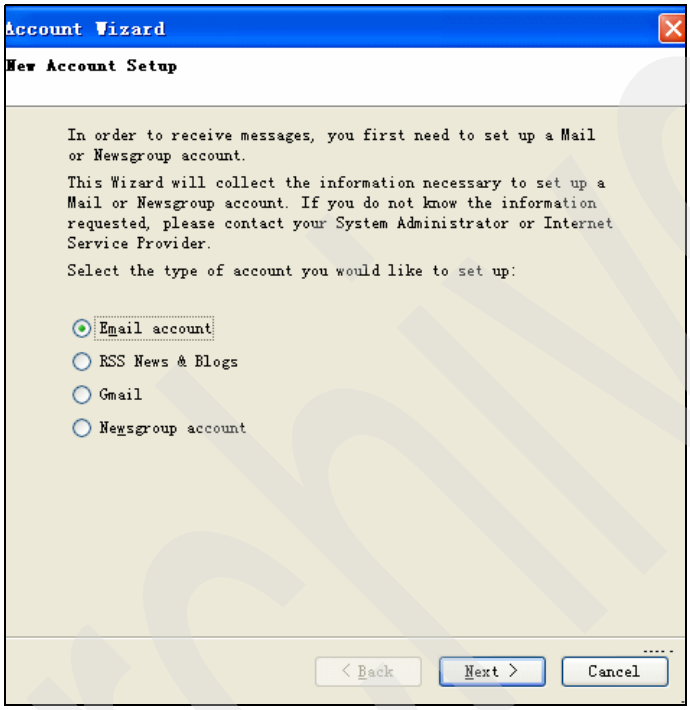


Figure 7-10 First panel of the Mozilla Thunderbird account wizard

The second panel in the account wizard asks for your name and e-mail address, as seen in Figure 7-11 on page 223.

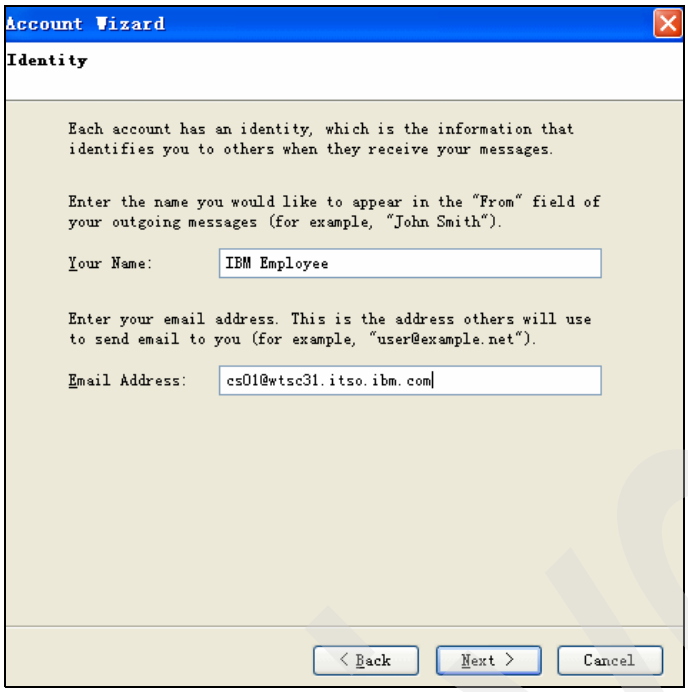


Figure 7-11 Second panel of the Mozilla Thunderbird account wizard

The third panel in the account wizard asks for the type of incoming server, that server's host name, and the outgoing server host name. Popper is the application responsible for transferring incoming mail and it uses the POP3 protocol, so we select a server type of POP. The incoming and outgoing servers are the same in our case because we are running popper and sendmail on the same LPAR. Our settings for the third panel are shown in Figure 7-12.

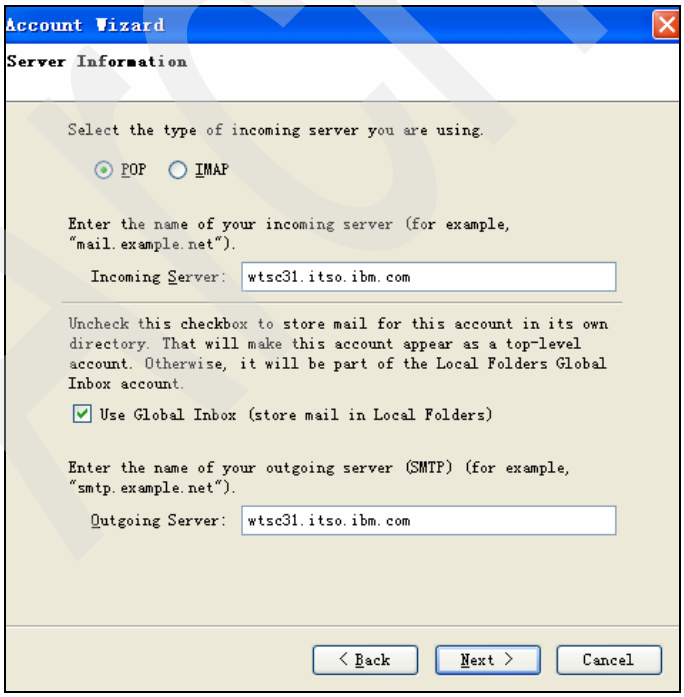


Figure 7-12 Third screen of the Mozilla Thunderbird account wizard

The fourth screen in the account wizard asks for the user ID to present to the incoming and outgoing servers. Again, because we are using the same LPAR for both popper and sendmail, there is only one user ID. Our user ID of cs02 is shown in Figure 7-13.

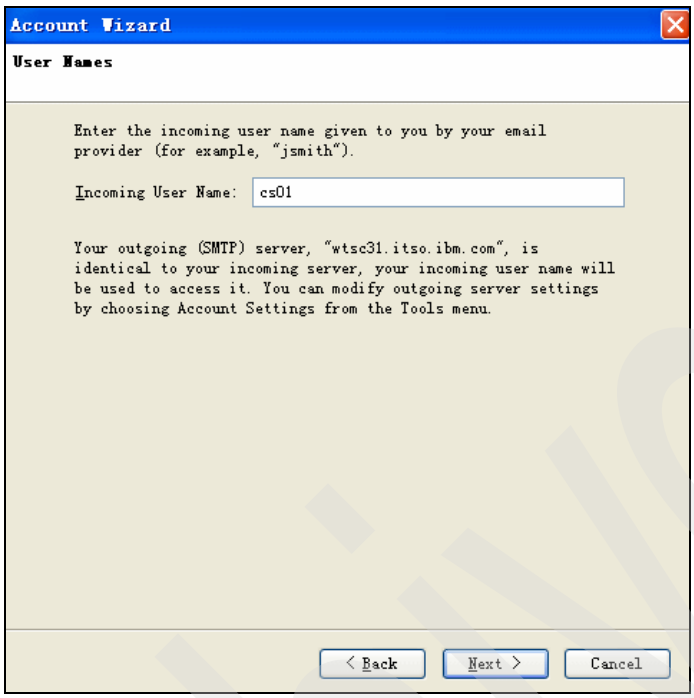


Figure 7-13 Fourth screen of the Mozilla Thunderbird account wizard

The fifth screen of the wizard asks for you to name the account. We used the name shown in Figure 7-14 on page 224.

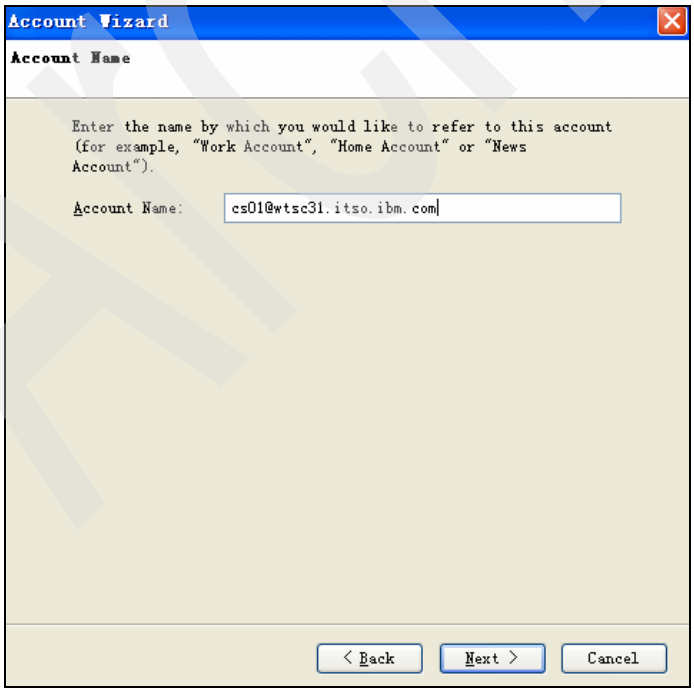


Figure 7-14 Fifth screen of the Mozilla Thunderbird account wizard

The final panel in the account wizard asks for you to confirm the settings entered in the previous panels. Our confirmation screen is shown in Figure 7-15.

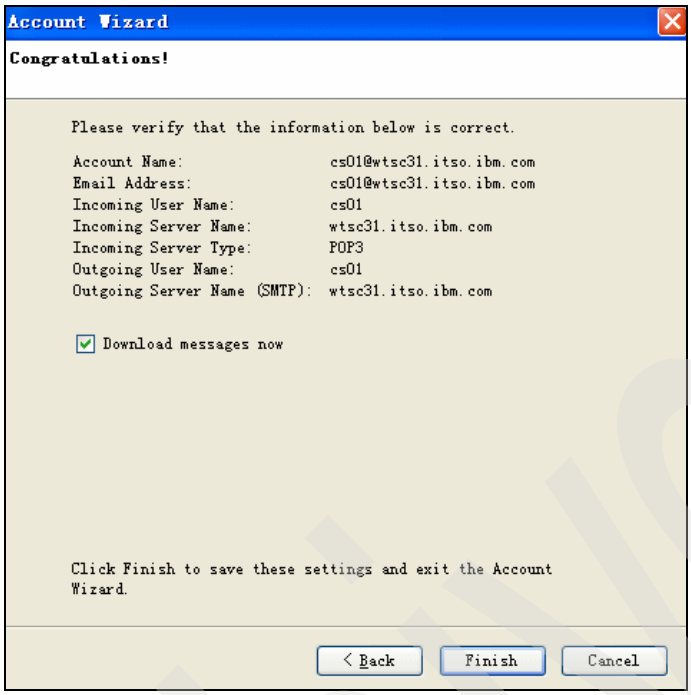


Figure 7-15 Final panel of the Mozilla Thunderbird account wizard

Now that Mozilla Thunderbird was configured to communicate with our system running sendmail, we sent a test message to ourself to test the environment. Figure 7-16 shows the message we sent.

Archived

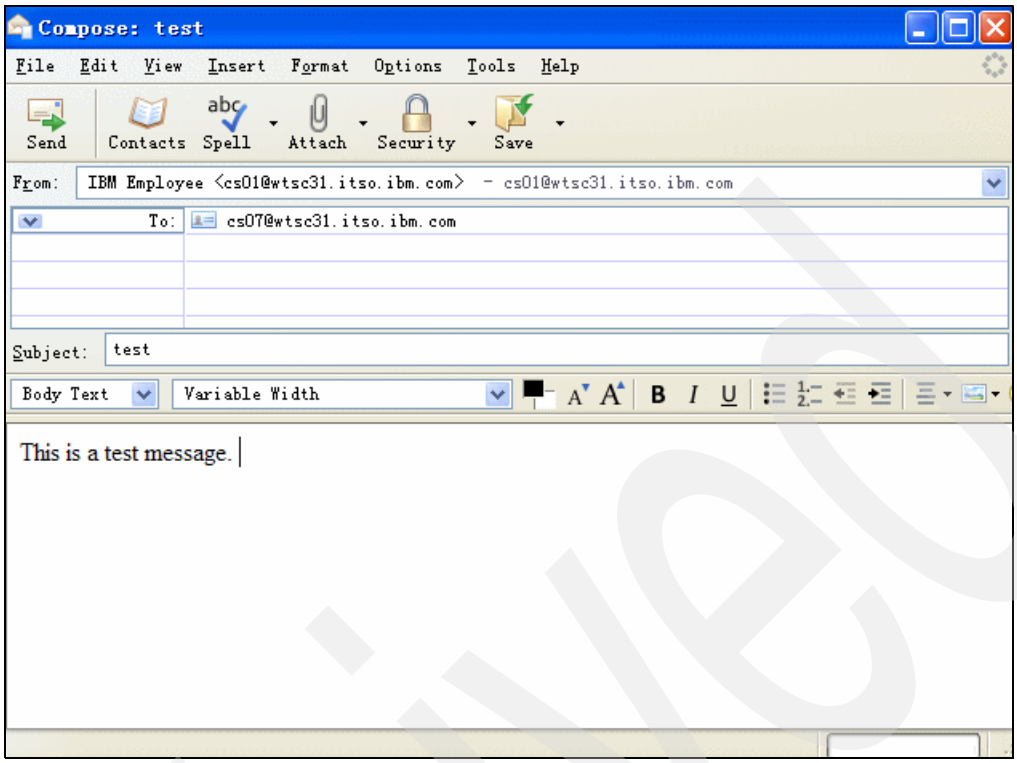


Figure 7-16 Outgoing test message

After a brief wait of approximately 10 seconds, Thunderbird confirmed that the e-mail message had been sent. At this point we have shown that sendmail is working properly and queuing up messages for delivery. To confirm that the message was delivered properly and to test popper's ability to transfer the message to our Mozilla Thunderbird mail user agent, we need to check for new mail.

After a brief wait, we clicked the **Get Mail** button in Thunderbird and received the message shown in Figure 7-17 on page 227.

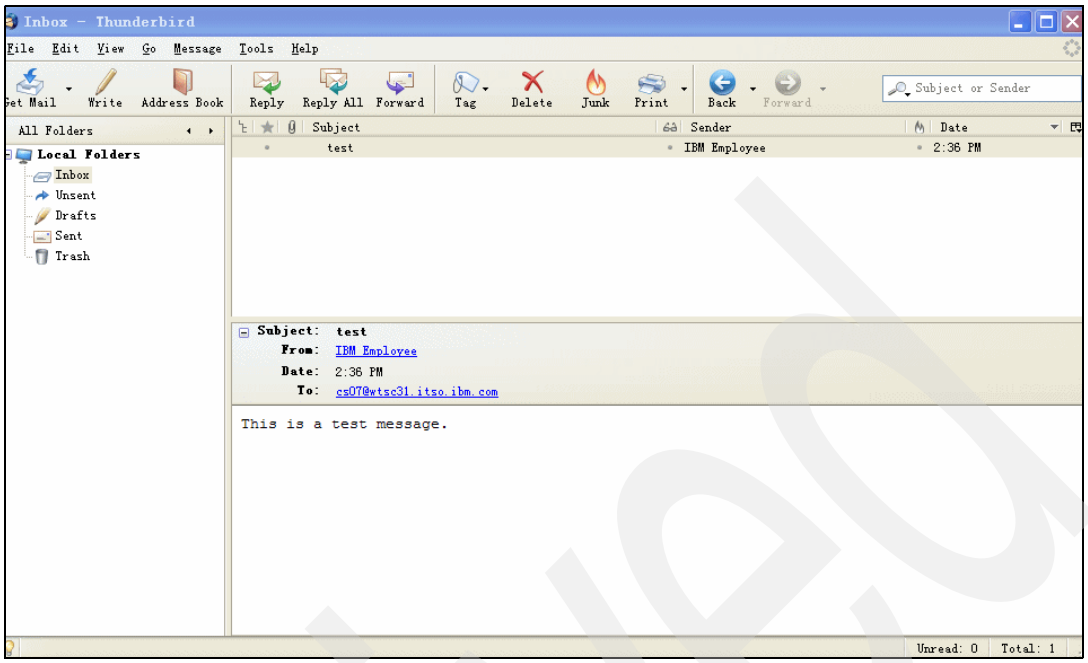


Figure 7-17 Inbound message

We have now shown that both sendmail and popper are up and running and able to deliver mail locally.

7.4 Using sendmail as a client

In this section we discuss the use of sendmail as an e-mail client. The topics are:

- Description of the sendmail client
- Configuration tasks for the sendmail client
- Verification of the sendmail client

7.4.1 Description of the sendmail client

Sendmail is also capable of acting as a client that can be used to send e-mail. Sendmail supports MIME attachments and also has built-in security features such as TLS/SSL sockets.

Sendmail can be very complex. Sendmail is incapable of interfacing with JES. If you want to submit batch job e-mail, use the SMTP batch client rather than the sendmail client. Refer to Figure 7-3 on page 201 to review the role of an SMTP client. Remember that a client can be an individual user or an SMTP command.

7.4.2 Configuration tasks for the sendmail client

We compose a message on z/OS and use sendmail to send it to another e-mail address. In this section, we also discuss how to create an e-mail attachment in our message. We attach a GIF image of the IBM logo to the bottom of our message.

The logo we use as an attachment is shown in Figure 7-18. We have FTPed the image file to the z/OS UNIX file system and placed it in /tmp/ibm-logo.gif.

Figure 7-18 GIF image of the IBM logo in /tmp/ibm-logo.gif

The configuration tasks for the sendmail client are:

- Set up sendmail and popper clients.
- Compose an RFC 822-compliant message.
- Encode the attachment.
- Join the encoded file with the text message.
- Add MIME attachment headers to the message.
- Invoke sendmail in the client mode to send the message.

Set up sendmail and popper clients

First, set up sendmail and popper as described in 7.3.3, “Verification of sendmail and popper setup” on page 221. The configuration file, alias file, and local-host-names files are all required for sendmail regardless of whether sendmail is run in server mode or client mode.

Compose an RFC 822-compliant message

Compose an RFC 822-compliant message in a file in the z/OS UNIX file system. RFC 822 specifies the format of a standard Internet e-mail message. You can view RFC 822 on the Internet at:

http://www.ietf.org/rfc/rfc822.txt

An RFC 822-compliant message is one that simply contains To:, From:, Subject:, and Date headers, followed by a blank line before the text of the message. We composed our RFC 822-compliant message in the z/OS UNIX file system in the /tmp/test-msg file. The contents of that file are shown in Example 7-12.

Example 7-12 RFC 822-compliant message in /tmp/test-msg

```
To: cs07@wtsc31.itso.ibm.com
From: cs01@wtsc31.itso.ibm.com
Date: Fri Sep 28 00:39:28 2007
Subject: test message
Hello,
   This is a test message using sendmail.
LGT
```

Encode the attachment

In order to send binary data via e-mail, you must encode the binary data into an e-mail-safe format. Modern e-mail applications use an encoding called base64 to encode binary files. We found the easiest way to perform base64 encoding was to download base64 encoding/decoding software from the Internet. We used the b64 program available at:

http://base64.sourceforge.neting

After downloading the source, we compiled the b64.c program from the z/OS UNIX shell using the command **cc -o b64 b64.c**.

> **Important:** Before compiling the b64.c program, change fprintf(outfile, \r\n) to fprintf(outfile, \n) in the b64.c file. This way, b64 will not append the unnecessary ^M characters.
>
> The ^M character represents an end-of-line character (carriage return) that is used in DOS-based systems, but is not used for end-of-line on UNIX-based systems. UNIX-based systems only use the line feed character to mark the end of a line.

Then we issued **b64 -e /tmp/ibm-logo.gif /tmp/ibm.gif** to encode the attachment. After encoding we used the command **cat /tmp/ibm.gif** to view the base64 encoded image. Example 7-13 shows the contents of our base64 encoded file.

Example 7-13 base64 encoded image in /tmp/ibm.gif

```
R0lGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR
zIuz3nzHtHyGVT0hYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

Join the encoded file with the text message

We can use the z/OS UNIX **cat** command to join our text message with our 7-bit ASCII image file:

```
cat /tmp/test-msg /tmp/ibm.gif > /tmp/test-message
```

The contents of the file /tmp/test-message are shown in Example 7-14.

Example 7-14 /tmp/test-message after joining the text and binary files

```
To: cs07@wtsc31.itso.ibm.com
From: cs01@wtsc31.itso.ibm.com
Date: Fri Sep 28 00:39:28 2007
Subject: test message
Hello,
   This is a test message using sendmail.
LGT
R0lGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR
zIuz3nzHtHyGVT0hYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

Add MIME attachment headers to the message

Multipurpose Internet Mail Extensions (MIME) are additional headers that can be included in e-mail messages. In e-mail messages, an attachment is simply binary data surrounded by MIME headers. A e-mail client capable of handling attachments simply looks for the MIME headers and separates the e-mail text from the binary attachment.

The headers required for our MIME message include the MIME-Version header and Content-Type header at the top of the message, with a subtype of multipart/mixed that identifies a unique boundary string for each part of the message. In each separate part of the message we added Content-Type and Content-Transfer-Encoding headers.

The headers we added to our message are shown in Example 7-15.

RFC 2045 and RFC 2046 describe MIME in full detail, and are available at:

http://www.ietf.org/rfc/rfc2045.txt and http://www.ietf.org/rfc/rfc2046.txt)

Example 7-15 /tmp/test-message with MIME headers

```
To: cs07@wtsc31.itso.ibm.com
From: cs01@wtsc31.itso.ibm.com
Date: Fri Sep 28 00:39:28 2007
Subject: test message
MIME-Version: 1.0
Content-Type: multipart/mixed;
 boundary="M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y"
This is a multi-part message in MIME format.
--M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y
Content-Type: text/plain
Content-Transfer-Encoding: 7bit

Hello,
   This is a test message using sendmail.
LGT

MWP
--M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y
Content-Type: image/gif;
 name="ibm.gif"
Content-Transfer-Encoding: base64
Content-Disposition: inline;
 filename="ibm.gif"

R0lGODlhLAAPAJEAAER3u////9Le7qK73SH5BAAAAAAALAAAAAAsAA8AAAJajB8iyAPAwntR
zIuz3nzHtHyGVT0hYnXqymKIa5jHxFBoi+fce4m9MRCldEQdj/b5SV6HYfG5OtokgBNSEgw4
mtBuJvZRiEYQXqzMLUO8xilDLHKgzOqyDVAAADsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
-M-U-L-T-I-P-A-R-T-B-O-U-N-D-R-Y--
```

Invoke sendmail in the client mode to send the message

At this point, we were able to send our message; we invoked sendmail with the following command:

```
/usr/sbin/sendmail -C /etc/mail/sendmail.cf -bm -t < /tmp/test-message
```

7.4.3 Verification of the sendmail client

To verify that our attachment was successfully sent, we simply checked our e-mail with Mozilla Thunderbird, which was set up as described in 7.3.3, “Verification of sendmail and popper setup” on page 221. After clicking **Get Mail** in Mozilla Thunderbird, we received the message shown in Figure 7-19.

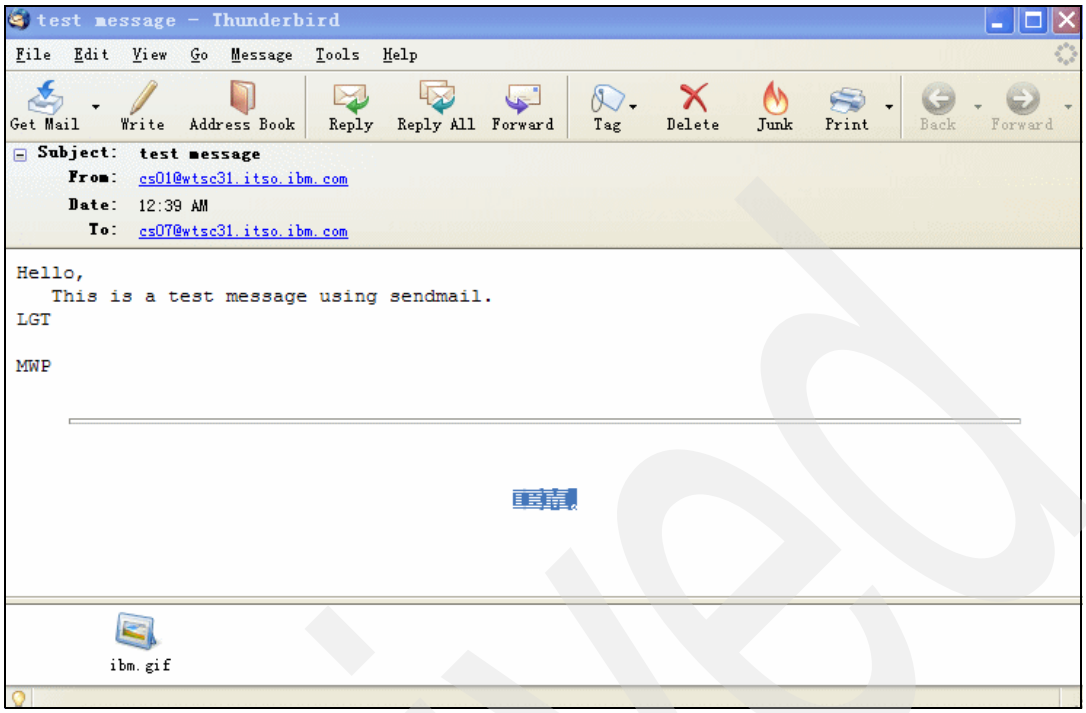


Figure 7-19 Message with attachment that was received from our z/OS system

7.5 Problem determination for the mail facilities

This section contains problem determination information for:

- Problem determination tasks for the z/OS SMTP server
- Problem determination for sendmail and popper
- Problem determination for the sendmail client

7.5.1 Problem determination tasks for the z/OS SMTP server

If changes to the domain name server require you to resolve already queued mail again, use the SMSG SMTP EXPIRE command, or the MODIFY SMSG,EXPIRE command. See the SMSGAUTHLIST statement for the SMTP server configuration data set for its usefulness in problem determination. You can also query operating statistics, such as mail delivery queues of the SMTP server. Some of the commands are shown in Table 7-1.

Table 7-1 Useful diagnostic commands for SMTP

TSO SMSG format	MVS MODIFY format
SMSG smtpproc HELP	F smtpproc,SMSG,HELP
SMSG smtpproc DEBUG or NODEBUG	F smtpproc,SMSG,DEBUG or NODEBUG
SMSG smtpproc TRACE or NOTRACE	F smtpproc,SMSG,TRACE or NOTRACE
SMSG smtpproc STARTEXIT or STOPEXIT	F smtpproc,SMSG,STARTEXIT or STOPEXIT
SMSG smtpproc STATS	F smtpproc,SMSG,STATS
SMSG smtpproc EXPIRE ipaddr	F smtpproc,SMSG,EXPIRE,ipaddr

Archived

TSO SMSG format	MVS MODIFY format
SMSG smtpproc NUMQUEUE	F smtpproc,SMSG,NUMQUEUE
SMSG smtpproc QUEUES	F smtpproc,SMSG,QUEUES
SMSG smtpproc SHUTDOWN	F smtpproc,SHUTDOWN

These administrative tasks are discussed in more detail in:

- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781

7.5.2 Problem determination for sendmail and popper

The sendmail "ba" book documents the detailed internal traces available in sendmail; refer to *sendmail 3rd Edition*, published by O'Reilly & Associates, Inc., ISBN 1-56592-839-3.

Popper accepts a command-line parameter -t, which specifies the location of a trace file for detailed tracing. Popper also logs messages to the *mail* syslogd facility.

Following are hints and tips for configuration checking when you meet problems on starting and using sendmail:

- SuperUser status is needed to start the sendmail daemon.
- The QueueDirectory option defined in the config file tells sendmail where to queue messages that are temporarily undeliverable. This directory must exist *before* sendmail is started.
- Sendmail is highly dependent on the Domain Name Server (DNS); it is important that the resolver be set up correctly to avoid unnecessary searching for a user.
- A program-controlled environment is necessary for sendmail to run in daemon mode when BPX.DAEMON is enabled, because many functions of sendmail (especially daemon functions) require it to change the user ID (UID) without prompting for a password.
- The daemon must be started by root, as usual. Table 7-2 shows the recommended security file permissions of files that sendmail might use.

Archived

Table 7-2 Sendmail permission table

Path	Type	Owner	Mode	Required or configurable
/	Directory	*root*	555 dr-xr-xr-x	Required
/usr	Directory	*root*	555 dr-xr-xr-x	Required
/usr/sbin	Directory	*root*	555 dr-xr-xr-x	Required
/usr/sbin/sendmail	File	*root*	755 -rwxr-xr-x	Required
/bin/sendmail	File	*smmsp*	755 -rwxr-xr-x	Configurable[a]
/etc/mail	Directory	*root*	555 dr-xr-xr-x	Configurable
/etc/mail/sendmail.cf	File	*root*	444 -r--r--r--	Configurable
/etc/mail/submit.cf	File	*root*	444 -r--r--r--	Configurable
/var/spool/mqueue	File	*sendmail*	700 -rwx------	Configurable
/var/spool/clientmqueue	File	*smmsp*	770 -rwxrwx---	Configurable

a. Used only with RACF program control systems

Note: When sendmail is attempting to canonify a host name, some broken name servers will return SERVFAIL (a temporary failure) on T_AAAA (IPv6) lookups. To allow sendmail to accept this behavior, ResolverOptions in the configuration file is set to WorkAroundBrokenAAAA by default.

If a system has thousands of users defined in the Users list, the administrator might consider enabling the UNIXMAP class. This increases the speed of the security checks performed by sendmail. APAR OW30858 provides details about what is needed to enable the UNIXMAP class.

7.5.3 Problem determination for the sendmail client

The sendmail “bat” book documents the detailed internal traces available in sendmai; refer to *sendmail 3rd Edition*, published by O'Reilly & Associates, Inc., ISBN 1-56592-839-3.

7.6 Additional information sources for mail servers

Detailed information about the mail servers and protocols can be found in the following documents:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User’s Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781
- *TCP/IP Tutorial and Technical Overview*, GG24-3376

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Archived

8

z/OS UNIX Telnet server

The z/OS UNIX Telnet server, also known as otelnetd, enables remote Telnet clients to log directly on to the z/OS UNIX System Services shell without having to log onto TSO. The facility provides an ASCII line mode terminal screen interface instead of the 3270 full-screen interface offered by TSO. This chapter focuses on the z/OS UNIX Telnet server functions that are available in the z/OS Communications Server.

This chapter uses the terms *otelnetd* and *z/OS UNIX Telnet server* to refer to the Telnet server that provides access to the z/OS UNIX System Services shell. Other manuals may also refer to the same application as *oTelnet*, or *Telnet daemon*.

This chapter discusses the following z/OS UNIX Telnet server topics.

Section	Topic
8.1, “Conceptual overview of otelnetd” on page 236	The basic concepts of the z/OS UNIX Telnet server.
8.2, “z/OS UNIX Telnet server implementation” on page 237	Key characteristics of the z/OS UNIX Telnet server and why it may be important in your environment.
8.3, “Problem determination for otelnetd” on page 243	Commonly implemented z/OS UNIX Telnet server design scenarios, their dependencies, advantages, considerations, and our recommendations.
8.4, “Additional information sources for otelnetd” on page 243	Selected implementation scenarios, tasks, configuration examples, and problem determination suggestions.

© Copyright IBM Corp. 2008. All rights reserved.

8.1 Conceptual overview of otelnetd

As illustrated in Figure 8-1, otelnetd, the z/OS UNIX Telnet server, is one of the standard applications provided with the z/OS Communications Server. It interfaces with the z/OS UNIX Sockets through C-Sockets, passing its packets in and out of the Logical and Physical File Systems.

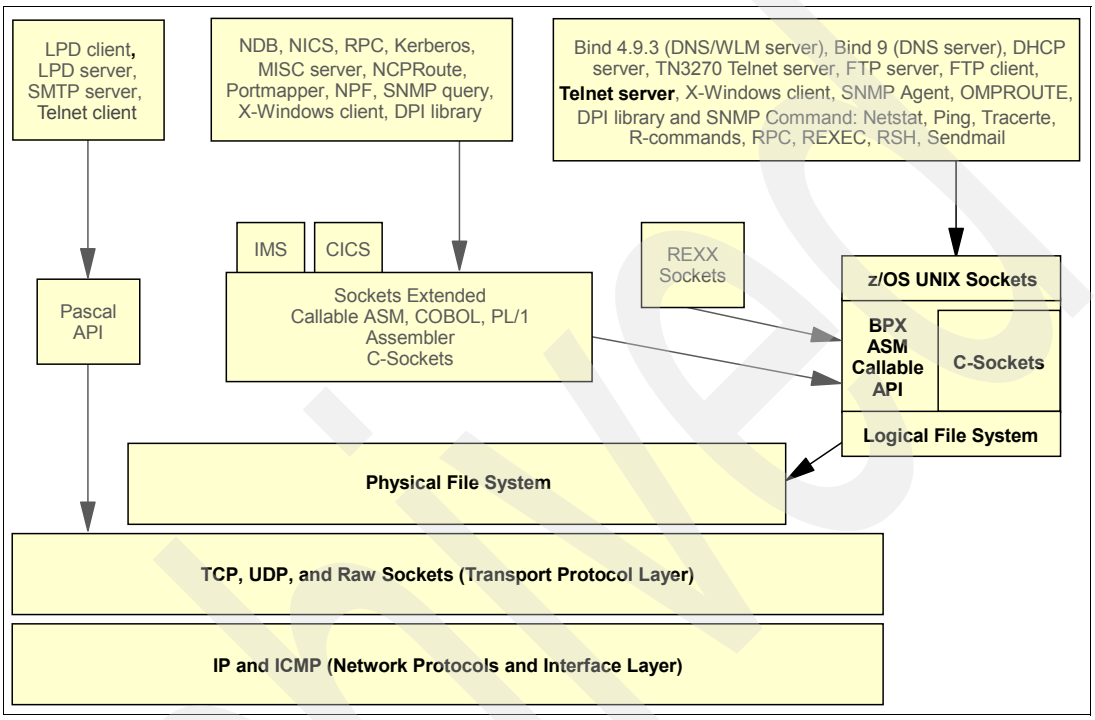


Figure 8-1 otelnetd application services

8.1.1 What is otelnetd?

The z/OS UNIX Telnet Server provides Telnet access directly to the z/OS UNIX shell, without requiring a TSO session. The z/OS Telnet server also provides different terminal capabilities. These features are very useful if you have UNIX applications that run under z/OS UNIX.

8.1.2 How does otelnetd work?

The z/OS UNIX Telnet server works in either character mode or line mode, but does not support SNA 3270 emulation (for 3270 support, see Chapter 2, “TN3270E Telnet server” on page 15). The z/OS UNIX server can optionally use Kerberos 5 authentication and DES encryption.

The z/OS UNIX Telnet server authenticates users, negotiates Telnet options with the clients, and then spawns a z/OS UNIX shell where z/OS UNIX commands can be executed. This process is illustrated in Figure 8-2.

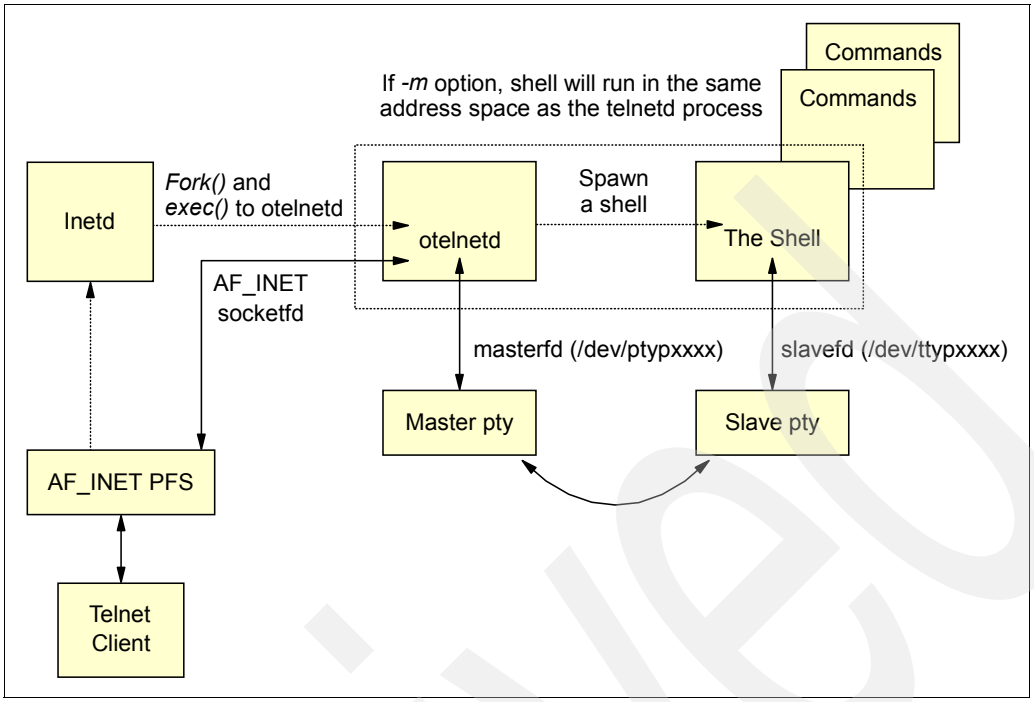


Figure 8-2 otelnetd interactions with INETD and z/OS UNIX

8.1.3 How can otelnetd be applied?

The z/OS UNIX Telnet server is a simple application with limited configuration options. We recommend running the z/OS UNIX Telnet server only if you require direct access to the z/OS UNIX shell. The z/OS UNIX Telnet server is only needed if you intend to access the z/OS UNIX shell without first accessing TSO or if you intend to use applications that depend on the special terminal capabilities available in z/OS UNIX.

The z/OS UNIX Telnet server is started by INETD, which is discussed in Chapter 6, “INETD” on page 185. Once INETD has created a TCP connection with a client, INETD forks and executes an instance of otelnetd.

8.2 z/OS UNIX Telnet server implementation

The relationships between otelnetd, the TN3270 Telnet server, and INETD are shown in Figure 8-3 on page 238.

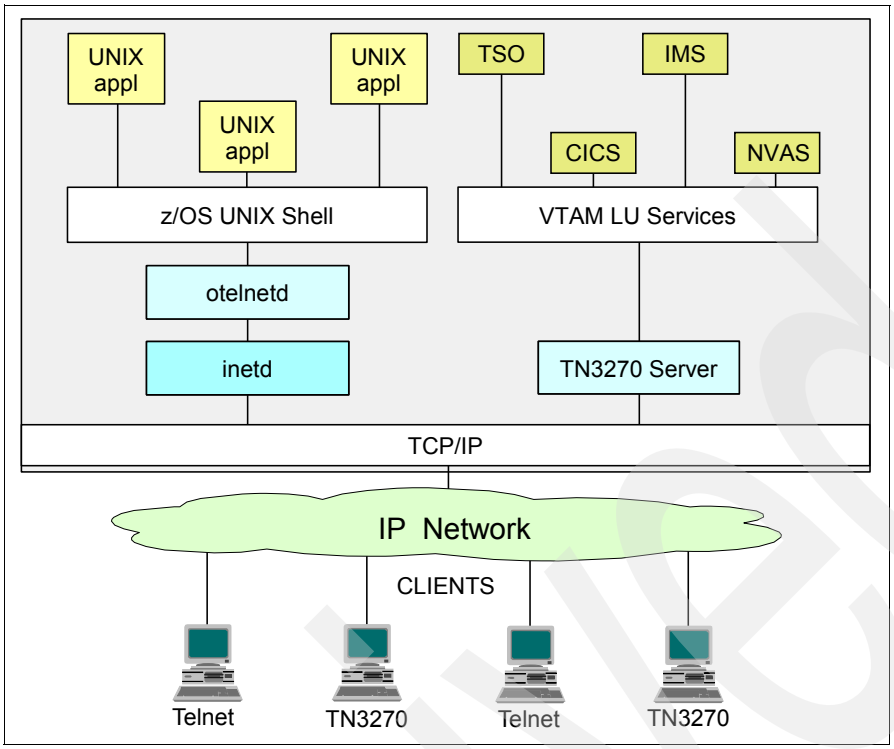


Figure 8-3 INETD and otelnet relationship

This section includes the following topics:

- Description of the otelnetd server
- Configuration tasks for otelnetd
- Activation and verification of otelnetd

8.2.1 Description of the otelnetd server

This section discusses otelnetd in a typical environment, with no authentication or encryption. The z/OS UNIX Telnet server is easy to implement and provides a quick method to access the z/OS UNIX shell. The z/OS Telnet server requires an active TCP/IP stack and INETD. We also highly recommended that you start syslogd to manage messages that may be generated by the otelnetd server.

> **Note:** The configuration we show may send sensitive data over insecure communication channels. If authentication and security is vital to your environment, then Kerberos security or AT-TLS provide options to secure the z/OS UNIX Telnet session. Kerberos is discussed in the *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926. AT-TLS is discussed in *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535.

8.2.2 Configuration tasks for otelnetd

In order to start the z/OS UNIX Telnet server, INETD must be active and contain a line in its configuration file for otelnetd. A port must also be reserved in /etc/services with the same service name.

The configuration tasks are:

- Plan the otelnetd environment
- Reserve a PORT in PROFILE.TCPIP
- Create a port entry in /etc/services
- Update the INETD configuration file for otelnetd

Note: If you have already set up and started INETD per our recommendations in Chapter 6, “INETD” on page 185, then you have already completed the necessary configuration and otelnetd may already be active.

Plan the otelnetd environment

Since z/OS is a native EBCDIC system, and the Telnet protocol is a native ASCII protocol, a method to perform ASCII/EBCDIC translation is required. The ASCII/EBCDIC translation is performed by otelnetd and relies on the **chcp** shell command to provide code page conversions. The **chcp** command supports both single-byte character set (SBCS) and double-byte character set (DBCS) code pages.

When a **chcp** shell command is executed, the otelnetd process is informed of the code page change via urgent data over the pseudo terminal connection (the master/slave pty interface). If a user selects to change the code page, the **chcp** command could be executed as part of the user's login process, for example, via the user's $HOME/.profile or $HOME/.setup shell scripts.

For more information about the **chcp** command, refer to *z/OS UNIX System Services Command Reference*, SA22-7802.

Consider the following topics before implementing otelnetd:

- Banner pages
- Pseudoterminals
- The terminal capabilities database
- Kerberos

Banner pages

An environment-specific banner page can be created in the file /etc/banner (shown in Example 8-1) using standard edit functions. It will be displayed at the client workstation after the user ID and password have been entered.

Example 8-1 File containing the banner page

```
************************************************************************
*                                                                      *
*  Welcome to z/OS UNIX System Services                                *
*                                                                      *
*  This is the /etc/banner page for Telnet server use.                 *
*                                                                      *
************************************************************************
```

With the /etc/banner file above, the welcome page in Example 8-2 was displayed on our terminal.

Example 8-2 Telnet welcome page

```
************************************************************************
*                                                                      *
*  Welcome to z/OS UNIX System Services                                *
*                                                                      *
*  This is the /etc/banner page for Telnet server use.                 *
```

```
*                                                                    *
**********************************************************************

IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2007
(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.
CS01 @ SC31:/u/cs01>
```

Pseudoterminals

Pseudoterminal files are created in the z/OS UNIX file system, in the /dev directory. The file names are ptyp*nnnn* and ttyp*nnnn*, where *nnnn* is a number from 0000 to 9999. These files are used in pairs by the otelnetd server. z/OZ UNIX creates pseudoterminal pairs dynamically as needed. In the rare case where the number of pseudoterminals needs to be increased, you can manually issue the **mknod** command to create additional pseudoterminals. The maximum number of pseudoterminals is constrained by the MAXPTYS statement in your BPXPRMxx parmlib member. For more information regarding pseudoterminals and the MAXPTYS parameter refer to *z/OS UNIX System Services Planning*, GA22-7800.

The terminal capabilities database

For telnet sessions, you may need some definitions of terminal capabilities in order to let z/OS UNIX application programs manipulate the terminal output correctly. These definitions are kept in the terminfo database.

A UNIX system, like other operating systems, must have at least one terminal attached to it. In the very early days these were typewriters having keyboard input and a paper output. These terminals were later replaced by video display units (VDUs), which behave like the typewriter did earlier—the paper is just replaced by the screen. These VDUs understand a data stream that contains ASCII characters to be printed, including control characters such as carriage return or new line. Differences appeared when individual manufacturers started to add specific commands to their terminals. Specific commands, for example, may include cursor movements (up, down, left, right, and so forth).

When one uses simple printing commands, such as **cat**, there is no obvious difference among terminals. However, these differences must be taken into account as soon as one runs a program that uses special commands. This is mostly the case with editors, such as vi or emacs. These programs need to know, for example, how to move the cursor to a specific location.

z/OS UNIX and other UNIX systems that have their roots in System V UNIX create a database named terminfo that contains definitions of all of the capabilities of each terminal. The terminfo database is shipped as part of z/OS. The database is populated with the terminal types defined by ibm.ti, dec.ti, wyse.ti, ansi.ti, and dtterm.ti. The database is in the directory /usr/share/lib/terminfo and the source files are in /samples. Each type of terminal that is defined has a corresponding file with the suffix .ti. If you need to recreate the terminfo database, run the **tic** utility. For example, to define an IBM terminal for the terminfo database, specify from the shell environment:

```
tic /samples/ibm.ti
```

To define terminal types such as VT100 and VT220, specify from the shell environment:

```
tic /samples/dec.ti
```

Table 8-1 shows the directories and files shipped as part of UNIX System Services or created by the **tic** command.

Table 8-1 Terminfo directory structure and contents

Directory	Terminal definition files
a	aixterm, aixterm-m, aixterm-m-old, aixterm-old
c	cdef
d	dtterm, dumb
h	hft, hft-c-old, hft-m-old, hft-nam-old, hft-c, hft-m, hft-nam, hft-old
i	ibmpc, ibmpcc, ibmpdp, ibm3101, ibm3151, ibm3151-noc, ibm3151-S, ibm3151-132, ibm3151-25, ibm3151-51, ibm3151-61, ibm3152, ibm3152-PS2, ibm3152-132, ibm3152-25, ibm3161, ibm3161-C, ibm3162, ibm3162-132, ibm3163, ibm3164, ibm5081, ibm5081-old, ibm5081-113, ibm5081-56, ibm5151, ibm5154, ibm5550, ibm5570, ibm6153, ibm6153-40, ibm6153-90, ibm6154, ibm6154-40, ibm6154-90, ibm6155, ibm6155-113, ibm6155-56, ibm8503, ibm8507, ibm8512, ibm8513, ibm8514, ibm8515, ibm8604
j	jaixterm, jaixterm-m
l	lft, lft-pc850
L	LFT, LFT-PC850
v	vs100, vs100s, vt100, vt100-am, vt100-nam, vt100x, vt200, vt200-8, vt320, vt320, vt320-8, vt330,vt330-8, vt340
w	wyse100, wyse30, wyse350, wyse50, wyse50-2, wyse60, wyse60-AT, wyse60-PC, wyse60-316X, wy100, wy30, wy350, wy50, wy50-2, wy60, wy60-AT, wy60-PC, wy60-316X
x	xterm, xterms

Whenever a shell environment is created, the terminal type of the client user is registered in the environment variable TERM. The TSO OMVS command environment handles TERM as though it were set to TERM=dumb. In a telnet session the TERM environment variable is set to the terminal type that the telnet client uses or emulates.

In order to use terminal types that are not supported by CS for z/OS IP you can create a directory to store local terminal definitions and use the TERMINFO environment variable to define the location of that directory. This environment variable was implemented to allow typical UNIX clients with GUI windowed command-line prompts access to the UNIX System Services Telnet server. This variable should be used if there are installation defined terminfo definitions. The environment variable can be passed directly to otelnetd with the -T command-line option, and should be set in /etc/profile or in the user's login script. Refer to the *z/OS UNIX System Services Planning*, GA22-7800, for more information about terminal definitions.

If you are interested in more information about termcap and terminfo, refer to *termcap & terminfo*, published by O'Reilly & Associates.

Kerberos

The z/OS UNIX Telnet server includes support for Kerberos 5 authentication and encryption over IPv4 connections. Kerberos is not supported on IPv6 connections. The Kerberos support is provided by z/OS Security Server. Refer to *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926, for more information.

Reserve a PORT in PROFILE.TCPIP

While not required, we recommend that you reserve the port to be used for otelnetd in PROFILE.TCPIP, using the INETD job name. If you do not reserve a port, it is possible that some other application will bind to the ports normally reserved for otelnetd. Our PORT statement for otelnetd is shown in Example 8-3 and reserves the ports for job name INETD1. We also chose to bind otelnetd to a specific DVIPA address so that otelnet could listen on port 23 on one address and TN3270 could listen on port 23 on a different IP address.

Example 8-3 PORT statement in PROFILE.TCPIP

```
PORT
  23 OMVS BIND 10.1.9.21 ; OE Telnet Server
```

Create a port entry in /etc/services

An entry in /etc/services is required to map the otelnet service name to a port number. Our entry in /etc/services is shown in Example 8-4 and maps the service otelnet to port 23 tcp.

Example 8-4 Statement in /etc/services

```
otelnet         23/tcp
```

Update the INETD configuration file for otelnetd

An entry is required in the INETD configuration file so that INETD will set up a listening socket for otelnetd. The application otelnetd can accept a number of command-line parameters that are specified in the INETD configuration file. All of the possible command-line parameters are discussed in detail in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. We use only the -m option, which has the same affect as the _BPX_SHAREAS environment variable, allowing forked or spawned processes to coexist in the same address space. Our INETD configuration file entry is shown in Example 8-5.

Example 8-5 Statement in INETD configuration file

```
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
```

8.2.3 Activation and verification of otelnetd

Both z/OS UNIX Telnet and TN3270 listen on port 23 by default. There are two methods to run both servers concurrently:

- Start TN3270 on port 23, and assign a different port to z/OS UNIX Telnet. Any clients connecting to the z/OS UNIX Telnet server will need to specify the alternate port.
- Start TN3270 on port 23 on one IP address, and start z/OS UNIX Telnet on port 23 but bound to a different IP address. Any clients wanting to connect to the z/OS UNIX Telnet server will need to specify that different IP address or host name for the z/OS UNIX Telnet server.

Verification of otelnetd is simple: use your favorite Telnet client, connect to the address and port that otelnetd is listening on, and log in. If you can log in and everything is readable, then otelnetd is working correctly.

Figure 8-4 shows that we successfully logged in to our environment.

```
Telnet wtsc31oe.itso.ibm.com
EZYTE27I login: cs01
EZYTE28I cs01 Password:
*************************************************************************
*                                                                       *
*  Welcome to z/OS UNIX System Services                                 *
*                                                                       *
*  This is the /etc/banner page for Telnet server use.                  *
*                                                                       *
*************************************************************************

IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2007
(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.

CS01 @ SC31:/u/cs01>
```

Figure 8-4 Successful login

8.3 Problem determination for otelnetd

The z/OS UNIX Telnet server includes a detailed trace, enabled with the -t command-line parameter, and can display information about the Telnet session with the -D command line option. *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, provides a detailed example of the trace output generated by otelnetd.

8.4 Additional information sources for otelnetd

For additional information on otelnetd, refer to:

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

Tip: For descriptions of security considerations that affect specific servers or components, see “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Archived

Archived

9

Remote execution

Remote execution servers enable execution of commands that have been received from a remote client. These servers run the remote execution Command Daemon (REXECD), which supports both the remote execution (REXEC) protocol and the Remote Shell (RSH) protocol. This chapter focuses on the remote execution functions that are available in the z/OS Communications Server.

This chapter discusses the following topics.

Section	Topic
9.1, “Conceptual overview of remote execution” on page 246	The basic concepts of remote execution and remote shell.
9.2, “TSO remote execution server” on page 250	How to set up and use a remote execution server in the TSO environment.
9.3, “z/OS UNIX remote execution server” on page 257	How to set up and use a remote execution server in the UNIX environment.
9.4, “REXEC TSO client command using user ID/password” on page 262	How to use the TSO client REXEC command with a user ID and password on the command.
9.5, “REXEC TSO client command using the NETRC data set” on page 266	How to use the TSO client REXEC command when using the NETRC data set to provide the user ID and password.
9.6, “REXEC UNIX client command” on page 271	How to use the UNIX client `rexec` command.
9.7, “Problem determination for z/OS remote execution facilities” on page 274	Problem determination techniques for the TSO and UNIX environments for remote execution servers and clients.
9.8, “Additional information sources on remote execution and remote shell” on page 277	References to additional reading sources for remote execution.

© Copyright IBM Corp. 2008. All rights reserved.

9.1 Conceptual overview of remote execution

As illustrated in Figure 9-1, REXEC and RSH are standard applications provided with the z/OS Communications Server to support remote execution services. The remote execution utilities use z/OS UNIX sockets to pass data through the Logical and Physical File Systems.

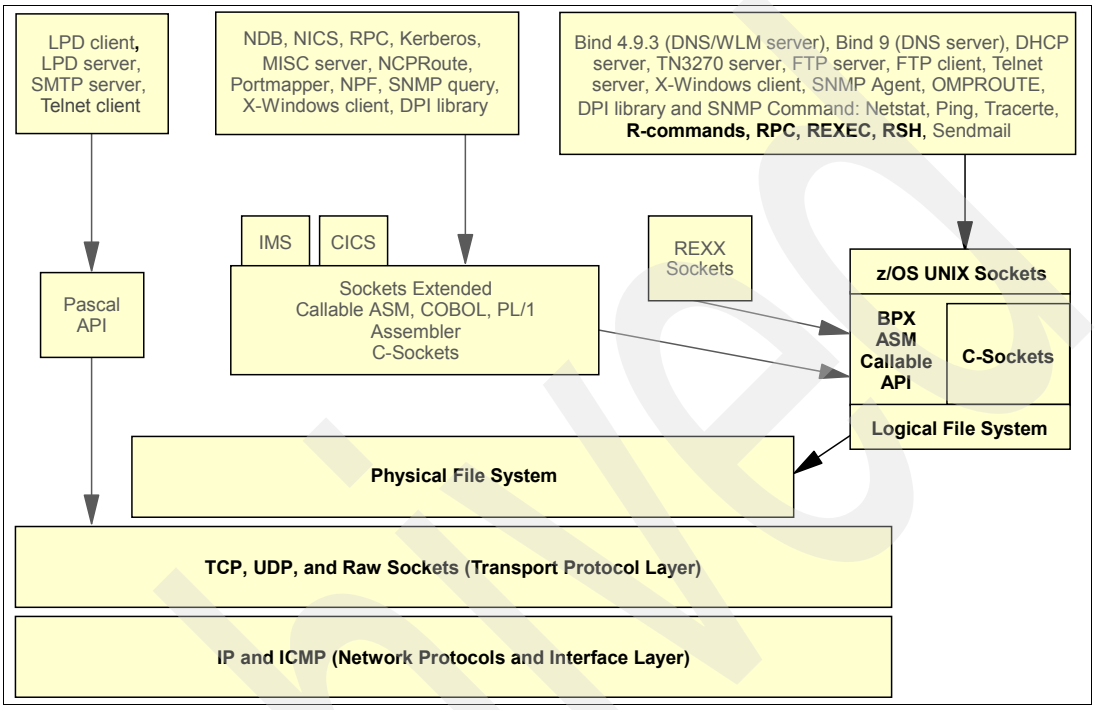


Figure 9-1 z/OS remote execution services

This section discusses the following topics:

- What is remote execution?
- How does remote execution work?
- How can remote execution be applied?

9.1.1 What is remote execution?

A client system may need to execute commands on another system without logging onto that target system directly. Remote execution is a simple solution for that requirement. Processes on one system in the IP network may need to initiate or post processes on another system to keep their environments synchronized. These processes might be automated, event driven, or manually activated by an operator. A system may not have the software necessary to communicate with another system. An operator may not have a terminal or terminal emulation software that is compatible with the other platform in order to log on directly to that target system. Remote execution is a command-level capability rather than an application-level one. Rather than enabling application-to-application communication, it establishes a low-level command submission and response retrieval process between two disparate systems that might otherwise not be able to communicate.

The z/OS Communications Server includes the following remote execution functionality:

- A TSO remote execution server (REXECD) and client (REXEC)
- A z/OS UNIX remote execution server (orexecd) and client (orexec)
- A z/OS UNIX remote shell server (orshd) and client (orsh)

- A TSO remote shell client

> **Note:** The TSO remote execution server (REXECD) also supports the RSH protocol. Therefore a separate TSO remote shell server is not included in the z/OS Communications Server.

Terminology

Terminology and acronyms can be very confusing. The documents referenced in 9.8, "Additional information sources on remote execution and remote shell" on page 277, are not necessarily consistent with their names and terminology when discussing the remote execution environments. We have consolidated the various names and terms associated with these two servers into a concise list to help you understand the references when you read them.

- The three terms *remote execution server*, *TSO remote execution server*, and *TSO RXSERVE daemon* refer to the single server *REXECD*, which is executed in the MVS environment as a started task.
  - REXECD is the distinguishing name for this TSO remote execution server.
  - SYS1.SEZAINST(RXPROC) is the sample JCL member for REXECD.
  - RXSERVE is the system proclib member name given to REXECD.
  - RXSERVE is the started task name used in the PROFILE.TCPIP data set.
  - The TSO RXSERVE daemon is the RXSERVE started task.
  - The RXSERVE task accepts and processes both commands, REXEC and RSH.
- The three terms *UNIX remote execution server*, *z/OS UNIX remote execution server*, and *z/OS UNIX REXECD server* refer to the single server *orexecd*, which is initiated by INETD in the UNIX environment.
  - *orexecd* is the distinguishing name for this z/OS UNIX remote execution server.
  - *rexecd* is considered a synonym of orexecd throughout the references.
  - orexecd resides in /usr/sbin/orexecd.
  - The orexecd server accepts and processes the **rexec**/**orexec** command.
- The three terms *Remote Shell Server*, *UNIX daemon*, and *z/OS UNIX RSHD* refer to the same server *orshd*, which is initiated by INETD in the UNIX environment.
  - *orshd* is the distinguishing name for this z/OS UNIX remote shell server.
  - *remote shell*, *rsh*, and *rshd* are all used in the manuals to refer to RSHD.
  - orshd resides in /usr/sbin/orshd.
  - The orshd server accepts and processes the **rsh**/**orsh** command.
- The term *remote execution protocol* generically refers to the support of the **rexec command**, whether discussing a server or client.
- The term *remote shell protocol* generically refers to the support of the **rsh command**, whether discussing a server or client.
- The client commands **REXEC** and **RSH** are used in the TSO environment, either in an interactive TSO session or in a batch TSO job.
- The client commands **rexec**, **orexec**, **rsh**, and **orsh** are used in the z/OS UNIX environment.
  - **rexec** and **orexec** are synonyms, and are used interchangeably.
  - **rsh** and **orsh** are synonyms, and are used interchangeably.

9.1.2 How does remote execution work?

Figure 9-2 on page 248 shows the relationship of the remote execution servers to the TCP/IP stack and the remote clients.

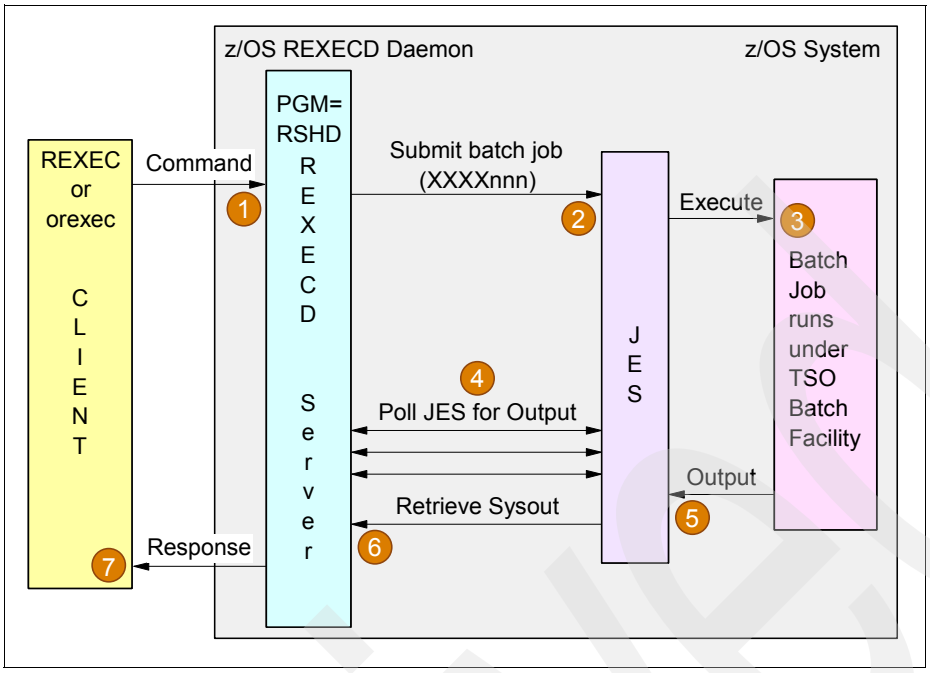


Figure 9-2 Remote execution client/server relationship

TSO remote execution server

The TSO remote execution *server* enables execution of TSO commands that have been received from a remote client. This server runs the remote execution Command Daemon (REXECD), which supports both the remote execution (REXEC) protocol and the Remote Shell (RSH) protocol.

TSO remote execution client

The TSO remote execution *client* is a TSO command that enables execution of a command on a remote system with the output returned to the TSO session. The TSO remote execution client can be run from TSO or in batch. When run from batch, the output of the remote command is returned to the batch job log. The remote host need not be a z/OS system, but can be any system with an REXEC server.

z/OS UNIX remote execution server

The z/OS UNIX remote execution server enables execution of z/OS UNIX commands that have been received from a remote client. Unlike the TSO remote execution server, this server runs just the remote execution Command Daemon (orexecd) supporting the remote execution (REXEC) protocol only.

z/OS UNIX remote execution client

The z/OS UNIX remote execution client, like the TSO remote execution client, enables execution of a command on a remote system with the output returned to your z/OS UNIX system services shell.

z/OS UNIX remote shell server

The z/OS UNIX remote shell server allows for execution of z/OS UNIX commands that have been received from a remote client. It is very similar to the z/OS UNIX remote execution server, but uses a different protocol.

z/OS UNIX remote shell client

The z/OS UNIX remote shell client, like the TSO remote execution client, enables execution of a command on a remote system with the output returned to your z/OS UNIX system services shell. The remote host need not be a z/OS system, but can be any system with a remote shell server.

TSO remote shell client

The TSO remote shell client enables execution of a command on a remote system with the output returned to your TSO session. The TSO remote shell can be executed from TSO and in batch. When run from batch, the output of the remote command is returned to the batch job log. The remote host need not be a z/OS system, but can be any system with a remote shell server.

9.1.3 How can remote execution be applied?

Two different REXEC remote execution servers can be configured for the z/OS environment. The REXEC client command can be executed in the TSO and batch environments and in the z/OS UNIX environment. The TSO and batch environments have the same dependencies and considerations. In the REXEC TSO environment, the client can provide a user ID and password on the actual REXEC command, or a NETRC data set containing records with the appropriate user ID and password information can be used to avoid placing the security information on the command line.

Servers

You will, of course, need to implement the appropriate remote execution server or the commands that you need to process: TSO remote execution servers for TSO commands and z/OS UNIX remote execution servers for z/OS UNIX commands. If you need access to both TSO and z/OS UNIX, then both servers can be implemented at the same time in one of two ways:

- Use different ports for the z/OS UNIX servers.
- Bind the UNIX servers to a specific dynamic VIPA.

Given the current emphasis that organizations are placing on security for their environments, you may consider a client's ability to submit requests without a password a security risk. We do not recommend implementing the support in RXSERVE for an optional password; rather, we recommend requiring the user ID and the password on the command. If, however, you are required to support the optional password approach for the remote client's RSH command, then refer to the following documents for the detailed steps to implement that support for RXSERVE:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Security Server RACF Security Administrator's Guide*, SC28-1915
- *z/OS Security Server RACF General User's Guide*, SC28-1917

Clients

Either the TSO or the z/OS UNIX clients may be used, regardless of whether TSO, z/OS UNIX, or another platform is the server. If, however, the remote host requires user identification and access control, then you must use the TSO clients if you want to use NETRC data sets or to submit jobs in batch. Depending on your environment, clients on your z/OS systems may need to use all three methods of remote command execution. We recommend that you become familiar with the setup and installation requirements for supporting all remote execution clients.

Roles and environments

Their roles and execution environments are positioned here:

- TSO remote execution server

  A client system may need to execute commands on another system without logging onto that target system directly. Remote execution establishes a low-level command submission and response retrieval process between two disparate systems that might otherwise not be able to communicate.

- z/OS UNIX remote execution server

  A client system may need to execute commands on another system without logging onto that target system directly. Remote execution establishes a low-level command submission and response-retrieval process between two disparate systems that might otherwise not be able to communicate.

  The z/OS UNIX remote execution servers can be configured to listen on any port, not just the well-known ports of 512 and 514. This provides some flexibility to your environment.

- REXEC TSO client command using user ID/password

  By specifying a user ID and password on the REXEC command itself, all of the information required by the client system and by the server system is located in one place. You avoid the clerical effort of maintaining the NETRC data set. Records in that data set must have up-to-date information related to targeted machine names, user IDs, and current passwords.

- REXEC TSO client command using the NETRC data set

  By omitting the user ID and password on the REXEC command, the plain text information is not visible on the command line being entered. You can take advantage of using a NETRC data set where you can store all the different user IDs and passwords you may have on various remote systems.

  These multiple user IDs and passwords can be maintained in one place, and the NETRC data set can be read/write protected so that only the owning user can view the contents.

- REXEC UNIX client command

  Some users of z/OS do all their work within the z/OS shell. For those users this method enables them to remain in the shell to execute the remote execution client command.

As part of any implementation effort, two appendixes in this book are beneficial in planning your work:

- Environment variables are categorized by application in Appendix A, “Environment variables” on page 311.
- Sample files for each application are listed in Appendix B, “Sample files provided with TCP/IP” on page 323.

9.2 TSO remote execution server

This section provides an overview of the TSO remote execution environment. The following topics are discussed:

- Description of TSO remote execution server
- Configuration tasks for TSO remote execution server
- Activation and Verification of TSO remote execution server

Archived

9.2.1 Description of TSO remote execution server

The TSO remote execution server (RXSERVE) enables execution of a TSO command that has been received from a remote client system that is submitted by either the **rexec** or the **rsh** command.

In our environment we use the TSO REXEC client to execute a TSO command on the TSO remote execution server on another system. We also use the TSO RSH client to execute the same command on the same remote server. We show the clients executing in both batch and from TSO.

Both clients allow user ID and passwords to be specified on the command line. In addition, the REXEC client also supports the use of a NETRC data set. To illustrate the use of the NETRC data set, we use REXEC with the NETRC data set. We use RSH to illustrate use of the command line to pass the user ID and password.

TSO remote execution server dependencies

To process rexec/rsh requests from remote clients under TSO, you must execute the TSO remote execution server (RXSERVE).

A CINET environment is one in which multiple TCP/IP stacks exist on the same system image. In the context of TSO remote execution, these stacks are called *transports*. The TSO remote execution server has affinity to a specific transport in a CINET environment, so you will need to configure and execute a unique instance of the RXSERVE server for each TCP/IP stack that requires TSO remote execution services. RXSERVE determines the stack name to which it should establish affinity from the setting of the TCPIPJOBNAME variable in the TCPIP.DATA file. This file is specified in the RXSERVE JCL on the //SYSTCPD DD statement.

RXSERVE supports REXEC and RSH requests only on their well-known ports: port 512 for REXEC and port 514 for RSH. These ports cannot be configured or modified.

RXSERVE must already be actively running when a client sends a command. Otherwise, the client connection request fails.

Considerations for using TSO remote execution server

The TSO remote execution server (RXSERVE) supports both REXEC and RSH commands from a client. It does not provide flexibility to configure the two servers differently. It listens on the well-known ports only (512 and 514) and cannot be modified.

If you need to run either of the z/OS UNIX servers (orexecd or orshd) concurrently with RXSERVE, they must be configured to use a port other than their well-known ports because RXSERVE cannot be configured to listen on different ports. However, for an alternate approach to this see “Concurrent execution for TSO and z/OS UNIX remote execution servers” on page 260.

Note: When sending an RSH request to this server, the remote client can issue the RSH command without specifying a password. This server can be configured to accept or reject this type of RSH client command syntax.

If your RXSERVE server is to accept the request without a password, then you must perform a number of additional security-related tasks to ensure proper authentication and authorization for the client's request.

To support this option, the RXSERVE server must have certain surrogate job submission privileges granted by the security program on your system. Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for the details of using the security server to define these authorizations.

9.2.2 Configuration tasks for TSO remote execution server

Perform the following tasks to implement the RXSERVE started task:

- Update the AUTOLOG and PORT statements
- Determine which remote client commands to support
- Optionally permit the RXSERVE task to be a surrogate job
- Update the RXSERVE cataloged procedure
- Create an optional user exit routine for RXSERVE
- Create one or more TSO batch procedures for RXSERVE
- Establish security access to JESSPOOL files

Update the AUTOLOG and PORT statements

If the TCP/IP stack is to start the RXSERVE task automatically when the stack initializes, include the RXSERVE name in the AUTOLOG statement in the PROFILE.TCPIP data set, as shown in Example 9-1.

Example 9-1 RXSERVE AUTOLOG statement

```
AUTOLOG
   RXSERVE ; Remote Execution Server
ENDAUTOLOG
```

Determine which remote client commands to support

Reserve port 512 for the REXEC protocol listener and port 514 for the RSH protocol listener, as shown in Example 9-2.

Example 9-2 RXSERVE port reservations

```
PORT
   512 TCP RXSERVE ; Remote Execution Server
   514 TCP RXSERVE ; Remote Execution Server
```

Optionally permit the RXSERVE task to be a surrogate job

The remote execution client can send commands to RXSERVE through port 512 by using the following methods:

- Sending the REXEC command
- Sending the RSH command with both user ID and password
- Sending the RSH command with user ID only

Our example does not support the third method. We require a password. The client commands could be as shown in Example 9-3.

Example 9-3 Client commands received by RXSERVE

```
REXEC -l userid -p password -s 512 wtsc31.itso.ibm.com listalc
RSH -l userid/password -s 514 wtsc31.itso.ibm.com listalc
RSH -l userid -s 514 wtsc31.itso.ibm.com listalc
```

The RSH command in third line without a password failed on our implementation with the message in Example 9-4, because the server did not have RACF surrogate authority defined.

Example 9-4 RSH command with missing password, error message

```
EZA4386E rshd: Permission denied.
```

Update the RXSERVE cataloged procedure

Copy the sample procedure from hlq.SEZAINST(RXPROC) to a system PROCLIB and name it RXSERVE. Modify it to meet your installation standards. Specify the parameters that RXSERVE uses to generate the batch TSO jobs that it submits.

Important: Specify two different output classes for MSGCLASS and TSCLASS. MSGCLASS must be a *held* class.

A sample RXSERVE procedure is shown in Example 9-5.

Example 9-5 Sample RXSERVE procedure

```
BROWSE    SYS1.PROCLIB(RXSERVE) - 01.01            Line 00000000 Col 001 080
//RXSERVE PROC MODULE='RSHD',
//          EXIT=REXECEXT,
//          TSOPROC=REXECTST,
//          MSGCLASS=X,
//          TSCLASS=L,
//          MAXCONN=512,
//          PREFIX=RSHD,
//          PURGE=N,
//          IPV6=N,
//          SECLABEL=N,
//          TRACE=(LOG,NOSEND)
//*
//RXSERVE EXEC PGM=&MODULE,PARM=('EX=&EXIT,TSO=&TSOPROC',
//             'MSG=&MSGCLASS,TSC=&TSCLASS',
//             'MAX=&MAXCONN,PRE=&PREFIX,TR=&TRACE',
//             'PUR=&PURGE,IPV6=&IPV6,SL=&SECLABEL'),
//             REGION=0M,TIME=1440
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB&SYSCLONE.),DISP=SHR
```

The RXSERVE parameters are explained in Example 9-6.

Example 9-6 RXSERVE procedure JCL parameters

```
EXIT=modulename
   Name of an exit routine to alter the JOB and EXEC JCL statements of the TSO batch jobs
   submitted in behalf of the remote client commands
```

```
TSOPROC=procname
   Name of the TSO batch proc to be submitted in behalf of the client request

MSGCLASS=x
   Output class goes on the JOB card of the submitted job.
   Must be a HELD class.

TSCLASS=x
   Output class for the //SYSTSPRT DD of the submitted job.
   Must be different from MSGCLASS

PURGE=x (Y or N)
   Should the submitted job's output be immediately purged when job completes?

IPV6=x (Y or N)
   Support communications with IPv6 addressing? Default is Y.

SECLABEL=x (Y or N)
   Add a security label to the job card of the submitted job?
   (Multilevel security related (MLS)).

TRACE=options
   LOG | NOLOG
   SEND | NOSEND
   CLIENT=clientid | ALLCLIENTS
   RESET

MAXCONN=nnnn
   Maximum number of open sockets at any one time. Each client requires 2, Minimum = 512

PREFIX=xxxx
   First four characters on the submitted JOB card, followed by a number between 1 and
   MAXCONN. (//XXXXn - //XXXXnnnn)
```

Create an optional user exit routine for RXSERVE

This is a optional step. The defaults that RXSERVE uses to submit the batch jobs, or even the values you specify using the JCL parameters, probably do not adhere to your installation's JCL standards. If that is the case, you must create a user exit that will adjust the parameters and values for the JOB and EXEC JCL statements. See the *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for a detailed description and sample copy of the user exit source. The source can be found in *hlq*.SEZAINST(RXUEXIT).

A sample RXUEXIT source code is shown in Example 9-7.

Example 9-7 REXECEXT routine source code

```
PARMS    DSECT
PTRINET  DC    F'0'                AF-INET or AF-INET6 socket addres    s
PTRJOBP  DC    F'0'                Job statement parameters
PTREXECP DC    F'0'                EXEC statement parameters
PTRJES   DC    F'0'                JES control buffer
*
BUFSIZE  EQU   1024                JOB statement buffer size
*
* RXUEXIT INIT   'REXECD add class parameter to JOB statement'
* *
RXUEXIT  CSECT                     Establish the RXUEXIT csect
RXUEXIT  AMODE 31
```

```
RXUEXIT  RMODE ANY
         USING RXUEXIT,12          Establish code addressability
         STM   14,12,12(13)        Save the caller's registers
         LR    12,15               Setup the local base register
         LR    2,1                 Parm pointer
         USING PARMS,2             Parameter addressability
         L     4,PTRJOBP           Job card parameters
         LR    5,4                 Start of buffer
         LA    6,1                 Scan 1 byte at a time
         LA    7,BUFSIZE(5)        First byte after buffer
         BCTR  7,0                 Last byte to scan
SCANLOOP EQU   *
         CLI   0(5),0              Is this string termination ?
         BE    GOTEND              Yes
         BXLE  5,6,SCANLOOP        Continue scan for term
* ---------------------------------------------------------------
* If string is not null terminated, return without altering
* ---------------------------------------------------------------
         B     RETURN              Should not happen.
GOTEND   EQU   *
         LR    6,5                 address of null byte
         SR    5,4                 L'job parameter statements
         LA    5,L'CLASS(5)        New parameter length
         CH    5,=AL2(BUFSIZE)     Do we exceed buffer size?
         BNH   LENOK               No, there is room enough
* ---------------------------------------------------------------
* String length would exceed buf size so return without altering
* ---------------------------------------------------------------
         B     RETURN              Return without modification
*
LENOK    EQU   *
         MVC   0(L_CLASS,6),CLASS  Move class statement to buff
         L     6,PTRJES            Get address of JES buffer
         MVC   0(L_JES2,6),JES2CNTL Move JES2 control to buffer
RETURN   EQU   *
         LM    14,12,12(13)        Restore the registers
         LA    15,0(0,0)           Load the return code
         BR    14                  Return
*
         LTORG
*
CLASS    DC    C',CLASS=L'         Class statement            1
         DC    X'00'               null termination byte
L_CLASS  EQU   *-CLASS
*
JES2CNTL DC    C'/*JOBPARM SYSAFF=SC31' JES2 system affinity  2
         DC    X'00'                    null termination byte
L_JES2   EQU   *-JES2CNTL
*
JES3CNTL DC    C'//*MAIN SYSTEM=(MAIN1)' JES3 main assignment
         DC    X'00'                     null termination byte
L_JES3   EQU   *-JES3CNTL
*
         END
```

1 The CLASS statement in user routine should be assigned to the same value as the TSCLASS parameter specified in RXSERVE start procedure.

2 Specify the system affinities if your environment require it. If you remote access a z/OS image in a sysplex and there are different multiple naming convention TCP/IP stacks, this parameter can let your remote access commands go to the right system, otherwise the connection to TCP/IP stack could fail.

> **Important:** Before assigning a HOLD class for the MSGCLASS in RXSERVE start procedure, use the JES2 command $D OUTCLASS to ensure that the OUTDISP of this outclass is OUTDISP=(HOLD,HOLD).
>
> Otherwise, the job output will not be returned to the remote user, and the user will receive a `Jobname not found` message when the job times out. In addition, the MSGCLASS cannot be modified by the MODIFY command.

The user exit should have the AMODE(31) and RMODE(24) attributes to provide addressability to the input parameters. Example 9-8 is a sample assembler and link-edit the RXUEXIT source code.

Example 9-8

```
//ASMLINK  JOB ,,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,
//             REGION=0M,MSGLEVEL=(1,1)
//ASM     EXEC PGM=ASMA90,PARM='DECK,NOOBJECT,LIST'
//SYSIN    DD  DISP=SHR,DSN=TCPIPB.TCPPARMS(RXUEXIT)
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//         DD  DISP=SHR,DSN=SYS1.AMODGEN
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPUNCH DD  DSN=&&LOADSET,UNIT=SYSDA,DISP=(,PASS),
//             SPACE=(400,(100,100,1)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//SYSPRINT DD  SYSOUT=*
//LKED    EXEC PGM=HEWL,COND=(4,LT,ASM),
//             PARM='XREF,AMODE=31,RMODE=24'
//SYSLMOD  DD  DSN=SYS1.COMMPLEX.LINKLIB(REXECEXT),DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
```

The SYSLMOD data set should have APF authorization and in LINKLST. After get return code ‘0’ from this job, refresh LLA before using this user exit routine.

Create one or more TSO batch procedures for RXSERVE

When using the RXSERVE server, the procedure specified in the TSOPROC parameter of the startup procedure for RXSERVE must have the //SYSTSPRT DD statement appearing before any other output DD specifications in the procedure. RXSERVE uses the JES interface to select the first file of the first output group from the job. For example, if the batch procedure specified is TSOPROC=REXECTST, Example 9-9 shows the correct placement of the //SYSTSPRT DD statement within the JCL.

Example 9-9 Sample TSOPROC procedure

```
BROWSE    SYS1.PROCLIB(REXECTST) - 01.03            Line 00000000 Col 001 080
//REXECTST PROC
//GENERAL  EXEC PGM=IKJEFT01,DYNAMNBR=90,TIME=1440
//STEPLIB  DD DSN=ISP.SISPLOAD,DISP=SHR
//SYSPROC  DD DSN=ESA.SYS1.CLIST,DISP=SHR
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB31),DISP=SHR
//SYSTSPRT DD TERM=TS,SYSOUT=*
```

```
//SYSPRINT DD TERM=TS,SYSOUT=*
//SYSTERM  DD TERM=TS,SYSOUT=*
//*
```

RXSERVE overrides the class specification for the //SYSTSPRT DD statement with the value specified on the TSCLASS parameter before it submits the job. RXSERVE also adds a //SYSTSIN DD statement to the TSOPROC procedure, where it includes the TSO command requested by the client on the original REXEC or RSH command. Refer to Example 9-3 on page 253 to see what a client request looks like.

Establish security access to JESSPOOL files

The submitted TSO commands normally generate output. This output is placed onto the JES spool through the //SYSTSPRT DD JCL statement. RXSERVE pulls this output from the JES spool and returns it to the requesting client host. If the user IDs associated with the batch jobs submitted by RXSERVE require JESSPOOL access authority, use the IBM RACF security server or equivalent System Authorization Facility (SAF) interface to set the required level of authority. Security details can be found in:

- *z/OS Security Server RACF Security Administrator's Guide*, SC28-1915
- *z/OS Security Server RACF General User's Guide*, SC28-1917

9.2.3 Activation and Verification of TSO remote execution server

RXSERVE can be started by an automations package, by a manual command (S RXSERVE), or by TCP/IP at initialization of the TCP/IP address space by way of the AUTOLOG statement within the PROFILE data set. When RXSERVE is started, the system issues the messages shown in Example 9-10.

Example 9-10 RXSERVE startup messages

```
$$HASP100 RXSERVEB ON STCINRDR
IEF695I START RXSERVEB WITH JOBNAME RXSERVEB IS ASSIGNED TO USER TCPIP GROUP TCPGRP
$HASP373 RXSERVE  STARTED
EZA4400I Trace options: LOG,SEND,ALLCLIENTS
EZA4404I Parameters: MSGCLASS=X,TSCLASS=L,TSOPROC=REXETEST,MAXCONN=512,EXIT=REXECEXT
EZA4423I Parameters: PREFIX=RSHD,PURGE=N,IPV6=N,SECLABEL=N
Established affinity with TCPIPB 1
EZA4414I rexecd: Initialization using JES=2 selected by Automatic Selection
descarray is at 13550160, size is 8 bytes
descarray has 512 entries, entry size is 788
```

1 The TCP/IP stack affinity is specified in TCPDATA on SYSTCPD DD statement.

9.3 z/OS UNIX remote execution server

This section provides an overview of the UNIX remote execution server (orexecd/rexecd) environment. The following topics are discussed:

- Description of z/OS UNIX remote execution server
- Configuration tasks for z/OS UNIX remote execution server
- Activation and verification of z/OS UNIX remote execution server

9.3.1 Description of z/OS UNIX remote execution server

The UNIX remote execution server (orexecd/rexecd) and remote shell server (orsh/rsh) enables execution of a z/OS UNIX command that has been received from a remote client host using the **rexec** or **rsh** command.

In our environment we use the z/OS orexec client to execute a z/OS UNIX command on the z/OS orexecd server on another system. We also use the z/OS UNIX orsh client to execute the same command on the same remote server.

z/OS UNIX remote execution dependencies

To process **orexec/rexec** requests from remote clients under z/OS UNIX, you must execute the z/OS UNIX remote execution server (orexecd/rexecd). Likewise, to process orsh/rsh requests from remote clients under z/OS UNIX, you must execute the z/OS UNIX remote shell server (orshd/rshd).

The z/OS UNIX servers are initiated through the INETD server and can be configured to use ports other than the well-known ports, 512 and 514. If INETD and the servers have been configured to serve different ports, the client must be aware of this and specify that specific port on the request.

INETD listens on the configured port on behalf of the z/OS UNIX remote execution servers. INETD initiates an instance of the server at the time a client request is received. The server is not activated ahead of time. INETD must be listening on the appropriate port, and the client must specify the same port—otherwise the client connection request fails.

For rexec requests, INETD listens on the port number that is specified for *exec* in the /etc/services file. For rsh requests, INETD listens on the port number that is specified for “shell”. Make sure the intended port numbers are specified.

Considerations for using z/OS UNIX remote execution

The z/OS UNIX remote execution server (orexecd/rexecd) services only the REXEC, **orexec**, and **rexec** commands from a client. In order to service RSH, **orsh**, and **rsh** clients, the RSH server (orshd/rshd) must also be executed.

If you need to run the z/OS UNIX server (orexecd/rexecd) concurrently with RXSERVE, the z/OS UNIX server must be configured to support a port other than its well-known port of 512 because RXSERVE cannot be modified. For details on running both servers concurrently, see “Concurrent execution for TSO and z/OS UNIX remote execution servers” on page 260.

The z/OS UNIX remote execution clients cannot use a NETRC data set, and jobs cannot be submitted in batch.

9.3.2 Configuration tasks for z/OS UNIX remote execution server

The z/OS UNIX System Services remote execution server, orexecd, enables UNIX commands to be submitted from a remote host and executed on the local z/OS system. The INETD server listens on the designated port on behalf of the orexecd server. When a remote request comes in on that port from a remote client host, INETD initiates an instance of the orexecd server to process the inbound request. The orexecd server is identified by the service name of exec in /etc/services. Place the **exec** command, along with appropriate parameters, into the /etc/inetd.conf file.

The orexecd record in the INETD /etc/inetd.conf file could look like one of the entries in Example 9-11, depending on the options that may be desired. See the *z/OS Communications*

Server: IP Configuration Guide, SC31-8775, for a complete description of all the parameters available to rexecd.

Example 9-11 Sample /etc/inetd.conf file records for rexecd

```
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd -LV
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd -s
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd -d -l -v -c -s
exec      stream tcp nowait BPXOINIT /usr/sbin/rexecd rexecd
```

In our scenario, the actual record for **rexecd** in an /etc/inetd.conf file is shown in Example 9-12.

Example 9-12 Sample /etc/inetd.conf file with orexecd/rexecd record

```
#=====================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
echo     stream tcp nowait OMVSKERN internal
discard  stream tcp nowait OMVSKERN internal
chargen  stream tcp nowait OMVSKERN internal
daytime  stream tcp nowait OMVSKERN internal
time     stream tcp nowait OMVSKERN internal
echo     dgram  udp wait   OMVSKERN internal
discard  dgram  udp wait   OMVSKERN internal
chargen  dgram  udp wait   OMVSKERN internal
daytime  dgram  udp wait   OMVSKERN internal
time     dgram  udp wait   OMVSKERN internal
pop3     stream tcp nowait bpxroot  /usr/sbin/popper popper
```

Make certain that the *exec* entry in /etc/services indicates the intended port number over which you want rexecd to process requests. INETD will listen on behalf of orexecd/rexecd on the port specified in /etc/services. The default well-known port for exec/rexecd/orexecd is 512. Notice the default port for shell/rsh/orsh is 514, as shown in Example 9-13.

Example 9-13 /etc/services entries for exec and shell

```
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
```

9.3.3 Activation and verification of z/OS UNIX remote execution server

When INETD is up and active, use a NETSTAT/**onetstat** command to verify that it is listening on the port specified for exec/rexecd/orexecd, as shown in Example 9-14.

Example 9-14 NETSTAT display shows INETD listening on port 512

```
User Id  Conn     State
-------  ----     -----
INETD1   00000040 LISTEN
  LOCAL SOCKET:   0.0.0.0..512
  FOREIGN SOCKET: 0.0.0.0..0
INETD1   00000040 LISTEN
  LOCAL SOCKET:   0.0.0.0..514
  FOREIGN SOCKET: 0.0.0.0..0
```

Concurrent execution for TSO and z/OS UNIX remote execution servers

We previously mentioned that one way to concurrently execute both TSO and z/OS UNIX servers is to change the port numbers for the z/OS UNIX server in /etc/services. It could listen on a different port and not conflict with the TSO server. However, that would be confusing to remote clients that may require using the well-known port for submitting requests.

Another method can be used to achieve concurrent execution for the two servers. Both servers are generic listeners, or generic servers. We can make one of the servers appear to be a BIND-specific server and avoid the port conflict.

Because the remote execution servers are generic servers, they attempt to bind to INADDR_ANY when they are started. This allows them to listen on all defined IP addresses. However, this prevents both the TSO and z/OS UNIX remote execution servers from listening on the same port, and one of the servers would have to use a nonstandard port. Using the BIND parameter on the PORT reservation statement in the TCPIP profile data set allows both the TSO and z/OS UNIX remote execution servers to bind to the same ports using different IP addresses. The following steps illustrate how this can be done. For more information about the PORT reservation statement, the DEVICE and LINK statements, and the HOME statement, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

1. Define a VIPA address (either static or dynamic, but preferably dynamic) to the PROFILE.TCPIP data set with a pair of DEVICE and LINK statements (or VIPADYNAMIC statements) and add the LINK to the HOME statement for static VIPA. We used Dynamic VIPA and VIPARANGE statement in our setup. This is shown in Example 9-15.

Example 9-15 VIPA for BIND of `orexecd` and orsh

```
DEVICE VIPA1    VIRTUAL 0
 LINK  VIPA1L   VIRTUAL 0 VIPA1
;
   HOME
      10.1.4.21 IUTIQDF4L
      10.1.5.21 IUTIQDF5L
      10.1.6.21 IUTIQDF6L
      10.1.1.20 VIPA1L
      10.1.2.21 OSA2080L
      10.1.2.22 OSA20A0L
      10.1.3.21 OSA20C0L
      10.1.3.22 OSA20E0L
   ;
OR
   VIPADYNAMIC
     ;-------------------------------------------------------------------
```

```
    ;  Set aside some addresses for use with BIND and SIOCSVIPA IOCTL    -
    ;              (10.1.9.21 thru 10.1.9.24)                            -
    ;-----------------------------------------------------------------
     VIPARANGE  DEFINE  255.255.255.255 10.1.9.21
     VIPARANGE  DEFINE  255.255.255.255 10.1.9.22
     VIPARANGE  DEFINE  255.255.255.255 10.1.9.23
     VIPARANGE  DEFINE  255.255.255.255 10.1.9.24
  ENDVIPADYNAMIC
```

2. Add PORT statements to the TCPIP profile for both the TSO and z/OS UNIX remote execution servers. One of the servers will bind to the VIPA address. The other can bind to INADDR_ANY by not specifying the BIND parameter. In this case, the z/OS UNIX remote execution servers bind to the VIPA address, as shown in Example 9-16.

> **Important:** The server *with* the BIND parameter must be listed *before* the one without the BIND parameter. This setup directs all requests to ports 512 or 514 with a destination IP address of 10.1.9.22 to the z/OS UNIX remote execution servers.
>
> Requests to ports 512 or 514 with a destination IP address that is not 10.1.9.22 are directed to the TSO remote execution server.

Example 9-16 BIND-specific assignments for orexecd and orshd

```
PORT
...
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.1.9.21 ; OE Telnet Server D-VIPA
...
;  443 UDP HTTPS              ; http protocol over TLS/SSL
   512 TCP OMVS BIND 10.1.9.22           ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.1.9.22           ; UNIX RSHD   D-VIPA
   512 TCP RXSERVE                       ; TSO  REXECD
   514 TCP RXSERVE                       ; TSO  RSHD 515 TCP LPSERVEB
...
```

3. Verify in the /etc/services file that exec uses port 512 and shell uses 514.
4. Verify the setup by starting the stack with the new changes and starting RXSERVE and INETD, the two listeners involved.
5. Issue the NETSTAT command and it should now show that both REXECD servers are listening on port 512 and both RSH servers are listening on port 514. INETD is now listening for the z/OS UNIX remote execution servers, in this case, on the VIPA address only. This is shown in Example 9-17.

Example 9-17 NETSTAT verifies BIND-specific listeners

```
User Id  Conn     State
-------  ----     -----
INETD1   00000040 LISTEN
  Local Socket:   10.1.9.22..512
  Foreign Socket: 0.0.0.0..0
INETD1   0000009D LISTEN
  Local Socket:   10.1.9.22..514
  Foreign Socket: 0.0.0.0..0
RXSERVE  000000D4 LISTEN
  LOCAL SOCKET:   0.0.0.0..512
  FOREIGN SOCKET: 0.0.0.0..0
RXSERVE  000000D5 LISTEN
```

```
LOCAL SOCKET:   0.0.0.0..514
FOREIGN SOCKET: 0.0.0.0..0
```

9.4 REXEC TSO client command using user ID/password

This section provides an overview of the REXEC TSO client command with a user ID and password specified. The following topics are discussed:

- Description of REXEC TSO with user ID and password
- Configuration of REXEC TSO with user ID and password
- Verification of REXEC TSO with user ID and password

9.4.1 Description of REXEC TSO with user ID and password

The client can specify a user ID and password on the REXEC command line. This user ID and password pair is the user ID that the remote system is to use for authorizing execution of the requested command.

Dependencies of REXEC TSO with user ID and password

To use REXEC, a REXEC daemon must be running on the remote host. The REXEC client passes the user name, password, and command to the remote REXEC daemon. The daemon provides automatic logon and user authentication, depending on the parameters that you set.

The user ID specified on the REXEC command must be a valid user ID on the remote targeted system.

Considerations for using REXEC TSO with user ID and password

By specifying a user ID and password on the REXEC command itself, the information is transmitted along with the command to the remote system in plain text.

9.4.2 Configuration of REXEC TSO with user ID and password

This section shows how to use the REXEC TSO client command with a user ID and password specified. Use the REXEC command to execute a command on a remote host and receive the results on the local system, as shown in Example 9-18.

Example 9-18 REXEC TSO client command format

```
REXEC ? -d -n -l userid -p password -s portnum -t translate_file remote_host command
```

The *remote_host* and the *command* are both required. All other parameters are optional. However, because this example is discussing the use of a user ID and password on the REXEC command, we assume that those two parameters are included on the command for this example. The remote-host can be a DNS name to be resolved by DNS services, or it can be the IP address of the remote host.

The parameters -d, -l, -p, and -s are case sensitive and must be entered in lowercase letters. The user ID and password values may be case sensitive, depending on the remote operating system requirements.

Use a question mark (**?**) to obtain command help, as shown in Example 9-19.

Example 9-19 REXEC TSO client command help

```
===> REXEC ?
 rexec: no command given.
Usage: rexec -? -d -b <tab> -l <usr> -p <pwd> -n -s <port> -t <fn> -b <tab> Fhost cmd
  options: -
           -?          display this message.
           -d          turn on debug tracing.
           -b <tab>    specifies tab value,
                       valid range 1 - 12 (default 1).
           -l <usr>    specifies remote userid.
           -p <pwd>    specifies remote password.
           -n          prevent automatic login.
           -s <port>   specifies server port (default 512).
           -t <fn>     specifies translation table name.

  Example: rexec  -d -l guest -p guest hostname ls
```

Use **-d** to activate debug tracing.

Use -n to prevent an automatic logon, and to force a password prompt to the client. This option is applicable only when the NETRC data set has been used to obtain the user ID and password. It prevents the client support function from automatically using the obtained password, and forces a password prompt to the client. This example does not consider the -n parameter because we are assuming the use of a user ID and password on the REXEC command.

Use **-l** to specify the user ID to be used by the remote host.

Use **-p** to specify the password for the user ID on the remote host.

Use **-s** to specify the TCP port number of the REXEC server on the remote host. The port number specified here must match the port number on which the remote REXEC server is listening. The default well-known port is 512.

Use **-t** to override the default translation table name. A search order process is used by the system to determine which translation table to use, as shown in Example 9-20.

Example 9-20 REXEC TSO client command translation table search order

```
If no translation table name is specified by using the -t parameter, search for:
   userid.STANDARD.TCPXLBIN
   hlq.STANDARD.TCPXLBIN where hlq is specified by DATASETPREFIX in the TCPDATA file

If the standard table is not present, then a hardcoded default table identical to
hlq.SEZATCPX(STANDARD) is used.

If -t specifies a translation table of -t translate_file, search for:
   userid.translate_file.TCPXLBIN
   hlq.translate_file.TCPXLBIN

If the specified file is not found, REXEC terminates with message EZA4805I
```

REXEC TSO requests can be submitted from the TSO command line or from a batch job. These methods include:

- Submitting REXEC TSO requests including a user ID and password
- Submitting REXEC TSO requests in batch with a user ID and password

Submitting REXEC TSO requests including a user ID and password

Use the REXEC command with a user ID and password specified, requiring no NETRC data set, as shown in Example 9-21.

Example 9-21 REXEC TSO client command with a user ID and password

```
READY
rexec -l CS07 -p CS07PWSD 10.1.1.20 netstat home
```

Submitting REXEC TSO requests in batch with a user ID and password

You usually run REXEC interactively by entering the command and then receiving the results at your terminal. However, you can also run REXEC as a batch job. To accomplish this, you must supply the necessary job control language (JCL) and submit it to the Job Entry Subsystem (JES) using the TSO SUBMIT command. The command format when submitted as a batch job is the same as the command format for TSO, described in Example 9-18 on page 262.

The command can be entered as a parameter on the EXEC JCL statement, as shown in Example 9-22.

Example 9-22 Batch REXEC TSO client command via EXEC PARM=

```
BROWSE    CS07.TCPPARMS(JOBRX1) - 01.04             Line 00000000 Col 001 080
//CS07Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS07
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='-l CS07 -p CS07PWSD 10.1.1.20 netstat home'
//SYSPRINT DD DSN=CS07.NETSTAT.HOME,DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB31)
```

9.4.3 Verification of REXEC TSO with user ID and password

If you omit the user ID, the password, or both when entering the REXEC command, *and* if the remote host is not specified with a user ID or password in the NETRC data set, then the local system prompts the client for the missing parameters.

Note: The data set containing the JCL cannot have sequence numbers.

You can verify the results of the REXEC TSO client command by reviewing the response you get back on your TSO session, as shown in Example 9-23.

Example 9-23 Results of REXEC TSO client command from a TSO session

```
Home address list:
LinkName:   IUTIQDF4L
  Address:  10.1.4.21
    Flags:
LinkName:   IUTIQDF5L
  Address:  10.1.5.21
    Flags:
LinkName:   IUTIQDF6L
  Address:  10.1.6.21
    Flags:
LinkName:   VIPA1L
  Address:  10.1.1.20
    Flags:  Primary
LinkName:   OSA2080L
  Address:  10.1.2.21
```

```
    Flags:
LinkName:   OSA20A0L
  Address:  10.1.2.22
    Flags:
***
LinkName:   OSA20C0L
  Address:  10.1.3.21
    Flags:
LinkName:   OSA20E0L
  Address:  10.1.3.22
    Flags:
.......................
***
```

You can verify the results of the batch job REXEC TSO client command by reviewing the response it places in the output data set specified in the batch JCL, as shown in Example 9-24.

Example 9-24 Results of REXEC TSO client command from a batch job

```
BROWSE    CS07.NETSTAT.HOME                          Line 00000000 Col 001 080
READY
netstat home
Home address list:
LinkName:   IUTIQDF4L
  Address:  10.1.4.21
    Flags:
LinkName:   IUTIQDF5L
  Address:  10.1.5.21
    Flags:
LinkName:   IUTIQDF6L
  Address:  10.1.6.21
    Flags:
LinkName:   VIPA1L
  Address:  10.1.1.20
    Flags:  Primary
LinkName:   OSA2080L
  Address:  10.1.2.21
    Flags:
LinkName:   OSA20A0L
  Address:  10.1.2.22
    Flags:
LinkName:   OSA20C0L
  Address:  10.1.3.21
    Flags:
LinkName:   OSA20E0L
  Address:  10.1.3.22
    Flags:
.......................
READY
END
```

Archived

9.5 REXEC TSO client command using the NETRC data set

This section provides an overview of the REXEC TSO client command with a NETRC data set used to obtain user ID and password information. The following topics are discussed:

- Description of REXEC TSO client using NETRC
- Configuration of REXEC TSO client using NETRC
- Verification of REXEC TSO client using NETRC

9.5.1 Description of REXEC TSO client using NETRC

The client can omit the user ID and password information from the REXEC command line. The client function of TSO locates a NETRC data set, and retrieves the user ID and password information from an appropriate record within that data set. The user ID that is located is passed on the remote system for authorizing execution of the requested command.

Dependencies of REXEC TSO client using NETRC

To use REXEC, a REXEC daemon must be running on the remote host. The REXEC client passes the user name, password, and command to the remote REXEC daemon. The daemon provides automatic logon and user authentication, depending on the parameters that you set. The user ID specified on the REXEC command must be a valid user ID on the remote targeted system.

A NETRC data set must exist and be configured with appropriate machine, user ID, and password information.

Considerations for REXEC TSO client using NETRC

The NETRC data set must be maintained and kept up to date with current passwords for the user on each targeted machine involved.

If the NETRC data set is needed because the user ID and password have been omitted from the client command, and it is not found by the client support function, the client request fails.

9.5.2 Configuration of REXEC TSO client using NETRC

This section shows how to use the REXEC TSO client command, omitting the user ID and password from the command and acquiring them from the NETRC data set. Use the REXEC (TSO client) command to execute a command on a remote host and receive the results on the local system, as shown in Example 9-25.

Example 9-25 REXEC command format with no user ID or password: for use with NETRC data set

```
REXEC ? -d -n -s portnum -t translate_file remote_host command
```

REXEC TSO requests can be submitted from the TSO command line or from a batch job. These methods include:

- Submitting REXEC TSO client requests without a user ID and password
- Submitting REXEC TSO client requests in batch without user ID and password
- Preparing and using the NETRC data set

Submitting REXEC TSO client requests without a user ID and password

Use the REXEC command without a user ID and password specified, requiring the NETRC data set, as shown in Example 9-23 on page 264.

Example 9-26 REXEC TSO client command without a user ID and password

```
READY
rexec 10.1.1.20 netstat conn
```

Submitting REXEC TSO client requests in batch without user ID and password

You usually run REXEC interactively by entering the command and then receiving the results at your terminal. However, you can also run REXEC as a batch job. To accomplish this, you must supply the necessary job control language (JCL) and submit it to the Job Entry Subsystem (JES) using the TSO SUBMIT command. The command format when submitted as a batch job is the same as the command format described in "Submitting REXEC TSO requests including a user ID and password" on page 264.

The command is entered as a parameter on the EXEC JCL statement. The results of the command executed on the remote host are stored on the local host and by default written to the //SYSPRINT DD allocation. The data set characteristics should be consistent with the output from the command you are executing at the remote host. This is shown in Example 9-27.

Example 9-27 Batch REXEC TSO client command without a user ID and password

```
BROWSE    CS07.TCPPARMS(JOBRX3) - 01.03              Line 00000000 Col 001 080
//CS07Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS07
//STEP1    EXEC PGM=REXEC,REGION=0M,
//     PARM='10.1.1.20 netstat home'
//SYSPRINT DD SYSOUT=*
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
```

Preparing and using the NETRC data set

The NETRC data set provides an alternative to specifying the user ID and password as REXEC parameters. The following NETRC discussion applies to the interactive TSO environment as well as to the batch job environment. It is clearer to use the batch job environment to illustrate the use of the NETRC data set. REXEC uses the following search order to find the NETRC data set:

1. //NETRC DD statement
2. *userid*.NETRC.DATA
3. *tso_prefix*.NETRC
4. *userid*.NETRC

Note: NETRC characteristics to be aware of are:

- If the password is specified using the -p parameter on the REXEC client command, no NETRC data set is used. Otherwise, the NETRC data set is used to acquire the password.
- If the password is omitted from both the REXEC client command and the machine record within the NETRC data set, the REXEC program prompts you for your current password. You must be running in an interactive TSO session.
- If you have submitted a batch job to execute the REXEC client command, and you omit the password from both the REXEC client command and from the machine record within NETRC, then the REXEC client command fails because it cannot prompt for a password in batch.

The format of the records within the NETRC data set is shown in Example 9-28.

Example 9-28 NETRC record syntax

```
machine remotehost login userid password userpswd
```

The keywords *machine*, *login*, and *password* must be specified in lowercase, exactly as they are spelled in the example. The remotehost specification can be its DNS name or its IP address. The user ID and password might be case sensitive at the remote host, and if supplied in the incorrect case, failure might occur when connecting to a REXEC server. The NETRC data set used in our scenarios is shown in Example 9-29.It is a RECFM=FB, LRECL=80 sequential data set.

Example 9-29 Sample NETRC data set: cs07.NETRC

```
MACHINE 127.0.0.1                    LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC30.ITSO.IBM.COM          LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC31.ITSO.IBM.COM          LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC32.ITSO.IBM.COM          LOGIN CS07     PASSWORD CS07PSWD
MACHINE WTSC33.ITSO.IBM.COM          LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.10                    LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.20                    LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.30                    LOGIN CS07     PASSWORD CS07PSWD
MACHINE 10.1.1.40                    LOGIN CS07     PASSWORD CS07PSWD
```

When running REXEC in batch, the user ID assigned to the batch job is used as the high-level qualifier in locating the default *userid*.NETRC.DATA or the *userid*.NETRC data set. Example 9-30 shows the use of the *userid*.NETRC.DATA data set containing the user ID and password for the user ID assigned to the batch job. The output is sent to the data set indicated on the //SYSPRINT DD statement.

Example 9-30 Batch REXEC TSO client command JCL using default NETRC data set

```
BROWSE    CS07.TCPPARMS(JOBRX3) - 01.03             Line 00000000 Col 001 080
//JOBRX3 JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=&SYSUID
//STEP1    EXEC PGM=REXEC,REGION=0M,
//     PARM='10.1.1.20 netstat conn'
//SYSPRINT DD SYSOUT=*
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
```

The //NETRC DD statement can be used in the batch job to override the default search order. Example 9-31 shows the use of the //NETRC DD statement in batch.

Example 9-31 Batch REXEC TSO client command JCL overriding default NETRC data set

```
//RXTEST JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=&SYSUID
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='10.1.1.20 netstat conn'
//SYSPRINT DD SYSOUT=*
//NETRC    DD   DSN=CS07.OVERRIDE.NETRC,DISP=SHR
```

9.5.3 Verification of REXEC TSO client using NETRC

If you omit the user ID, the password, or both when entering the REXEC command, *and* if the remote host is not specified with a user ID or password in the NETRC data set, then the local system prompts the client for the missing parameters.

Note: The data set containing the JCL cannot have sequence numbers.

You can verify the results of the REXEC TSO client command by reviewing the response you get back on your TSO session, as shown in Example 9-32.

Example 9-32 Results of REXEC TSO client command without user ID and password in TSO

```
User Id  Conn     State
-------  ----     -----
DCD1DIST 0000225A Listen
  Local Socket:   0.0.0.0..38241
  Foreign Socket: 0.0.0.0..0
DCD1DIST 00002257 Listen
  Local Socket:   0.0.0.0..38240
  Foreign Socket: 0.0.0.0..0
OMPB     00000026 Establsh
  Local Socket:   127.0.0.1..1026
  Foreign Socket: 127.0.0.1..1027
TCPIPB   00000017 Listen
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 0.0.0.0..0
TCPIPB   0000001D Establsh
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 127.0.0.1..1025
TCPIPB   00000025 Establsh
  Local Socket:   127.0.0.1..1027
***
Foreign Socket: 127.0.0.1..1026
TCPIPB   0000001C Establsh
  Local Socket:   127.0.0.1..1025
  Foreign Socket: 127.0.0.1..1024
TN3270B  00000034 Listen
  Local Socket:   ::..23
  Foreign Socket: ::..0
TN3270B  00000033 Listen
  Local Socket:   ::..992
  Foreign Socket: ::..0
TCPIPB   000023A9 UDP
  Local Socket:   ::..3512
  Foreign Socket: *..*
***
```

An example of the batch job that RXSERVE submits (using the TSOPROC specified in the RXSERVE parms) is shown in Example 9-33. The originating user job did not specify a user ID or password on the REXEC command, so the REXEC program uses the default NETRC data set because the //NETRC DD statement was not supplied in the originating JCL.

Example 9-33 REXEC using default NETRC when no user ID or password specified on command

```
SDSF OUTPUT DISPLAY CS07Z    JOB07179  DSID     2 LINE 0       COLUMNS 02- 81
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 0  --  N O D E

23.22.51 JOB07179 ---- THURSDAY,    21 Sep 2007 ----
23.22.51 JOB07179  IRR010I  USERID CS07     IS ASSIGNED TO THIS JOB.
23.23.25 JOB07179  ICH70001I CS07     LAST ACCESS AT 23:19:07 ON THURSDAY, September
23.23.25 JOB07179  $HASP373 CS07Z    STARTED - INIT 3    - CLASS C - SYS SC30
        1 //CS07Z    JOB  (TT,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS0
        2 //STEP1    EXEC PGM=REXEC,REGION=0M,
          //     PARM='-d 10.1.1.20 netstat home'
        3 //SYSPRINT DD SYSOUT=*
        4 //SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(DATAB30)
ICH70001I CS07     LAST ACCESS AT 23:19:07 ON THURSDAY, September 21, 2007
```

```
IEF236I ALLOC. FOR CS07Z STEP1
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I 803B ALLOCATED TO SYSTCPD
IEF237I 803B ALLOCATED TO SYS00004
Established affinity with TCPIPB
EZA4809I  Using NETRC file //'CS07.NETRC'.
EZA4762I  MACHINE : 10.1.1.20
EZA4763I  LOGIN   : CS07
EZA4764I  PASSWORD: ******
EZA4801I  MVS TCP/IP REXEC CS
EZA4775I  Calling function rexec_af with the following:
Host: 10.1.1.20   user: CS07   cmd: netstat home   port: 512
Getaddrinfo successful
EZA4774I  rexec invoked;
Data socket = 1  Control socket = 3
```

The job that was submitted by RXSERVE to execute the requested command under TSO batch is shown in Example 9-34.

Example 9-34 TSOPROC job submitted by RXSERVE (REXECTST)

```
SDSF OUTPUT DISPLAY CS074    JOB07180  DSID     2 LINE 0       COLUMNS 02- 81
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 0  --  N O D E

23.22.57 JOB07180 ---- THURSDAY, 21 Sep 2007 ----
23.22.57 JOB07180  ICH70001I CS07     LAST ACCESS AT 23:22:57 ON THURSDAY, September
23.22.57 JOB07180  $HASP373 CS074    STARTED - INIT 2    - CLASS A - SYS SC30
23.22.57 JOB07180  -                                    ---------TIMINGS (M
23.22.57 JOB07180  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB   VECT
23.22.57 JOB07180  -CS074    TSO      GENERAL     00    466    .00    .00    .00
23.22.57 JOB07180  -CS074    ENDED.  NAME-                    TOTAL CPU TIME=
23.22.57 JOB07180  $HASP395 CS074    ENDED
------ JES2 JOB STATISTICS ------
  21 Sep 2007 JOB EXECUTION DATE
           11 CARDS READ
          108 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
            4 SYSOUT SPOOL KBYTES
         0.00 MINUTES EXECUTION TIME
        1 //CS074    JOB CS07,
          // USER=CS07,
          // PASSWORD=,
          // MSGCLASS=A
        2 //TSO EXEC REXECTST
        3 XXREXECTST PROC
        4 XXGENERAL  EXEC PGM=IKJEFT01,DYNAMNBR=90,TIME=1440
        5 XXSTEPLIB  DD DSN=ISP.SISPLOAD,DISP=SHR
        6 XXSYSPROC  DD DSN=ESA.SYS1.CLIST,DISP=SHR
        7 XXSYSTCPD  DD DSN=TCPIPB.TCPPARMS(DATAB30),DISP=SHR
        8 //SYSTSPRT DD SYSOUT=(H,RSHD),HOLD=YES
          X/SYSTSPRT DD TERM=TS,SYSOUT=*
        9 //SYSPRINT DD RECFM=VBA,SYSOUT=(*,RSHD),HOLD=YES
          X/SYSPRINT DD TERM=TS,SYSOUT=*
       10 //SYSTERM  DD RECFM=VBA,SYSOUT=(*,RSHD),HOLD=YES
          X/SYSTERM  DD TERM=TS,SYSOUT=*
          XX*
       11 //DEFAULT  OUTPUT DEFAULT=YES,CONTROL=PROGRAM,WRITER=RSHD
       12 //SYSTSIN  DD *
 STMT NO. MESSAGE
```

You can verify the results of the batch job REXEC TSO client command by reviewing the response it places into the output data, as shown in Example 9-35.

Example 9-35 Results of REXEC TSO client command without user ID and password in batch

```
BROWSE    CS07.NETSTAT.CONN                          Line 00000000 Col 001 080
User Id  Conn     State
-------  ----     -----
DCD1DIST 0000225A Listen
  Local Socket:   0.0.0.0..38241
  Foreign Socket: 0.0.0.0..0
DCD1DIST 00002257 Listen
  Local Socket:   0.0.0.0..38240
  Foreign Socket: 0.0.0.0..0
OMPB     00000026 Establsh
  Local Socket:   127.0.0.1..1026
  Foreign Socket: 127.0.0.1..1027
TCPIPB   00000017 Listen
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 0.0.0.0..0
TCPIPB   0000001D Establsh
  Local Socket:   127.0.0.1..1024
  Foreign Socket: 127.0.0.1..1025
TCPIPB   00000025 Establsh
  Local Socket:   127.0.0.1..1027
  Foreign Socket: 127.0.0.1..1026
TCPIPB   0000001C Establsh
  Local Socket:   127.0.0.1..1025
  Foreign Socket: 127.0.0.1..1024
TN3270B  00000034 Listen
  Local Socket:   ::..23
  Foreign Socket: ::..0
TN3270B  00000033 Listen
  Local Socket:   ::..992
  Foreign Socket: ::..0
TCPIPB   00002396 UDP
  Local Socket:   ::..3507
  Foreign Socket: *..*
```

9.6 REXEC UNIX client command

This section provides an overview of the REXEC UNIX client command. The following topics are discussed:

- Description of the REXEC UNIX client command
- Configuration of the REXEC UNIX client command
- Verification of the REXEC UNIX client command

9.6.1 Description of the REXEC UNIX client command

The client can use the z/OS UNIX shell to enter the REXEC UNIX form of the rexec command. The command synonym, **orexec**, can be used.

Dependencies of the REXEC UNIX client command

To use REXEC, a REXEC daemon must be running on the remote host. The REXEC client passes the user name, password, and command to the remote REXEC daemon. The daemon provides automatic logon and user authentication, depending on the parameters that

you set. The user ID specified on the REXEC command must be a valid user ID on the remote targeted system.

Considerations for using the REXEC UNIX client command

The UNIX REXEC client command does not provide the option of omitting the user ID and password. It cannot take advantage of the NETRC data set.

9.6.2 Configuration of the REXEC UNIX client command

Use the z/OS UNIX REXEC command (**rexec**/**orexec**) to execute a command on a remote host and receive the results on the local system, as shown in Example 9-36. The **rexec** command is a synonym for the **orexec** command in the z/OS UNIX shell. They both have the same format.

Example 9-36 REXEC UNIX client command format

```
orexec ? -d -l userid -p password -s portnum remote_host command
```

The parameters -d, -l, -p, and -s are case sensitive and must be entered in lowercase letters. The user ID and password values may be case sensitive, depending on the remote operating system requirements.

Use a question mark (?) to obtain command help, as shown in Example 9-37.

Example 9-37 REXEC UNIX client command help

```
CS01 @ SC31:/u/cs01>rexec
Usage: orexec|rexec -V -d -l <user> -p <pwd>
                -s <port> fhost command
       options: -
                -?        display this message
                -d        turn on debug tracing
                -l <usr>  specifies remote login id
                -p <pwd>  specifies remote password
                -s <port> specifies server port
                -C        Uppercase messages
                -V        display APAR level
                -e <wait> select time limit

Example: orexec -d -l guest -p guest hostname ls -l
CS01 @ SC31:/u/cs01>orexec
Usage: orexec|rexec -V -d -l <user> -p <pwd>
                -s <port> fhost command
       options: -
                -?        display this message
                -d        turn on debug tracing
                -l <usr>  specifies remote login id
                -p <pwd>  specifies remote password
                -s <port> specifies server port
                -C        Uppercase messages
                -V        display APAR level
                -e <wait> select time limit

Example: orexec -d -l guest -p guest hostname ls -l
CS01 @ SC31:/u/cs01>
```

Use -d to activate debug tracing.

Use -n to prevent an automatic logon, and to force a password prompt to the client. This option is applicable only when the NETRC data set has been used to obtain the user ID and password. It prevents the client support function from automatically using the obtained password, and forces a password prompt to the client. This example does not consider the -n parameter because we are assuming the use of a user ID and password on the REXEC command.

Use -l to specify the user ID to be used by the remote host.

Use -p to specify the password for the user ID on the remote host.

Use -s to specify the TCP port number of the REXEC server on the remote host.

Use *remote_host* to specify the remote host name or IP address to which you are sending the indicated command.

Use *command* to specify the command that is sent to the remote host. The command is composed of one or more words. The command you specify must not require a response from you in order to complete. The **rexec**/**orexec** command cannot interact with you after you enter data in the command format.

The port number specified in the **rexec**/**orexec** command must match the port number on which the remote REXEC server is listening. The default, if not specified here, is the port number specified on the *exec* entry within the */etc/services* file, as shown in Example 9-38.

Example 9-38 Sample exec entry in /etc/services

```
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
```

z/OS REXEC UNIX requests can be submitted from the z/OS UNIX shell command line. Use the **rexec**/**orexec** command with user ID and password specified, as shown in Example 9-39.

Example 9-39 REXEC UNIC client command with user ID and password

```
READY
orexec -l CS07 -p CS07PSWD -s 512 10.1.1.20 netstat home
```

9.6.3 Verification of the REXEC UNIX client command

You can verify the results of the REXEC UNIX client command by reviewing the response you get back, as shown in Example 9-40.

Example 9-40 Results of REXEC UNIX client command

```
CS01 @ SC31:/u/cs01>rexec -l CS07 -p CS07PSWD -s 512 10.1.1.20 netstat home
Home address list:
LinkName:   IUTIQDF4L
  Address:  10.1.4.21
    Flags:
```

```
LinkName:   IUTIQDF5L
  Address:  10.1.5.21
    Flags:
LinkName:   IUTIQDF6L
  Address:  10.1.6.21
    Flags:
LinkName:   VIPA1L
  Address:  10.1.1.20
    Flags:  Primary
LinkName:   OSA2080L
  Address:  10.1.2.21
    Flags:
LinkName:   OSA20A0L
  Address:  10.1.2.22
    Flags:
LinkName:   OSA20C0L
  Address:  10.1.3.21
    Flags:
LinkName:   OSA20E0L
  Address:  10.1.3.22
    Flags:
...
```

9.7 Problem determination for z/OS remote execution facilities

This section includes problem determination techniques for the remote execution servers and client commands in the z/OS TSO and z/OS UNIX environments. These techniques are covered in the following topics:

- Problem determination for TSO remote execution
- Problem determination for REXEC TSO with user ID and password
- Problem determination of REXEC TSO using NETRC
- Problem determination for the REXEC UNIX client command

9.7.1 Problem determination for TSO remote execution

You can use the MVS MODIFY command to change some of the RXSERVE server operating parameters during execution. The following parameters cannot be changed while RXSERVE is executing:

- IPV6
- SECLABEL
- MAXCONN
- PREFIX

When the MODIFY (F) command is used to modify the RXSERVE parameters, RXSERVE responds with messages similar to those shown in Example 9-41.

Example 9-41 Modifying RXSERVE parameters dynamically

```
F RXSERVEB,TSOPROC=$RISC,MSGCLASS=A,TSCLASS=Z,TRACE=LOG,EXIT=NOEXIT
EZA4400I Trace options: LOG,NOSEND,ALLCLIENTS
EZA4404I Parameters: MSGCLASS=A,TSCLASS=Z,TSOPROC=$RISC,MAXCONN=512,NOEXIT
```

9.7.2 Problem determination for REXEC TSO with user ID and password

When issuing a command to be executed on the remote host, do not place the command in quotation marks. The REXEC TSO client program may not process the command stream properly if quotes are included.

Submitting a long-running command may cause the REXEC program to end abnormally with a 522 system abend code. This can be avoided by specifying TIME=1440 on the EXEC JCL statement. Job step timing is suppressed.

If the command to be executed on the remote host contains a slash (/), you must use a preceding slash (/) in the PARM= string, as shown in Example 9-42.

Example 9-42 Executing a remote command having an imbedded slash (/): PARM=

```
//CS07C    JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS07
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='/-l oper6 -p cold2day 10.1.9.22 ls ./lpp/samples/*'
//SYSPRINT DD   DSN=CS07.LPP.SAMPLES.C,DISP=SHR
```

A condition code of 12 will be set by the system when an REXEC batch request encounters any of the following error conditions:

- The client program cannot connect to TCP/IP.
- The host name cannot be resolved.
- The translation table cannot be found or loaded.

The REXEC client command can fail if any of the following error conditions occur:

- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP may not allow the REXEC client program access to the stack.
  - REXEC may not be configured, or may be configured incorrectly.
- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- There is no REXEC server listening on the remote server.
  - You could be using an incorrect port number.
  - You could be using an incorrect server name or IP address.
- The remote system rejects the connect or submitted command.
  - The security may not be correctly set up for your user ID to log on to the remote server.
  - The security may not be correctly set up for your user ID to execute the requested command on the server.

9.7.3 Problem determination of REXEC TSO using NETRC

When issuing a command to be executed on the remote host, do not place the command in quotation marks. The REXEC TSO client program may not process the command stream properly.

Submitting a long-running command may cause the REXEC program to end abnormally with a 522 system abend code. This can be avoided by specifying TIME=1440 on the EXEC JCL statement. Job step timing is suppressed.

If the command to be executed on the remote host contains a slash (/), you must use a preceding slash (/) in the PARM= string, as shown in Example 9-43.

Example 9-43 Executing a remote command having an imbedded slash (/)

```
//CS07E JOB  (TST,0001),MSGLEVEL=(1,1),MSGCLASS=X,CLASS=C,NOTIFY=CS07
//STEP1    EXEC PGM=REXEC,REGION=0M,
//         PARM='/-l oper6 -p cold2day 10.1.9.22 ls ./lpp/samples/*'
//SYSPRINT DD   DSN=CS07.LPP.SAMPLES.E,DISP=SHR
```

The REXEC client command can fail if any of the following error conditions occur:

- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP may not allow the REXEC client program access to the stack.
  - REXEC may not be configured, or may be configured incorrectly.
- There is no REXEC server listening on the remote server.
  - You could be using an incorrect port number.
  - You could be using an incorrect server name or IP address.
- The remote system rejects the connect or submitted command.
  - The security may not be correctly set up for your user ID to log on to the remote server.
  - The security may not be correctly set up for your user ID to execute the requested command on the server.

9.7.4 Problem determination for the REXEC UNIX client command

The REXEC UNIX client command can fail if any of the following error conditions occur:

- The host name cannot be resolved.
  - The name could be spelled incorrectly.
  - The DNS server could be unavailable.
  - Use the IP address to see if access can be gained.
  - Ping the name to see if DNS can resolve the name and access the server.
  - Tracerte the name to see if network path access is an issue.
- The client program cannot connect to TCP/IP.
  - TCP/IP could be unavailable on the local or remote machine.
  - TCP/IP may not allow the REXEC client program access to the stack.
  - REXEC may not be configured, or may be configured incorrectly.
- There is no REXEC server listening on the remote server.
  - You could be using an incorrect port number.
  - You could be using an incorrect server name or IP address.
- The remote system rejects the connect or submitted command.

– The security may not be correctly set up for your user ID to log on to the remote server.
– The security may not be correctly set up for your user ID to execute the requested command on the server.

9.8 Additional information sources on remote execution and remote shell

Detailed information for the remote execution servers and protocols can be found in the following documents:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS UNIX System Services Planning*, GA22-7800

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

Archived

Archived

10

Domain Name Service

The Domain Name Service (DNS) is a client/server solution in which *name servers* provide information about host systems and their IP addresses. This chapter describes the z/OS DNS, which uses the Berkeley Internet Name Domain (BIND) software, the accepted standard of DNS. BIND was developed at the University of California, Berkeley, and is currently maintained by the Internet Software Consortium (ISC). This chapter focuses on the DNS BIND 9 functions that are available in the z/OS Communications Server.

This chapter discusses the following topics.

Section	Topic
10.1, “Conceptual overview of the DNS name server” on page 280	The basic concepts of Domain Name Service.
10.2, “Authoritative DNS server” on page 283	Key characteristics of an authoritative DNS server.
10.3, “Caching-only DNS server” on page 284	Key characteristics of a caching-only DNS server on z/OS, and how to configure and use it.
10.4, “Automated domain name registration” on page 294	How to configure and use ADNR facilities.
10.5, “Problem determination for DNS service” on page 304	Problem determination techniques for the DNS servers: the caching-only DNS and the ADNR.
10.6, “Additional information sources for DNS” on page 309	References to additional reading sources for the z/OS DNS server.

© Copyright IBM Corp. 2008. All rights reserved.

10.1 Conceptual overview of the DNS name server

As illustrated in Figure 10-1, the DNS name server is one of the standard applications provided with the z/OS Communications Server. The DNS server uses the z/OS UNIX Sockets interface to send and receive its packets through the Logical and Physical File Systems.

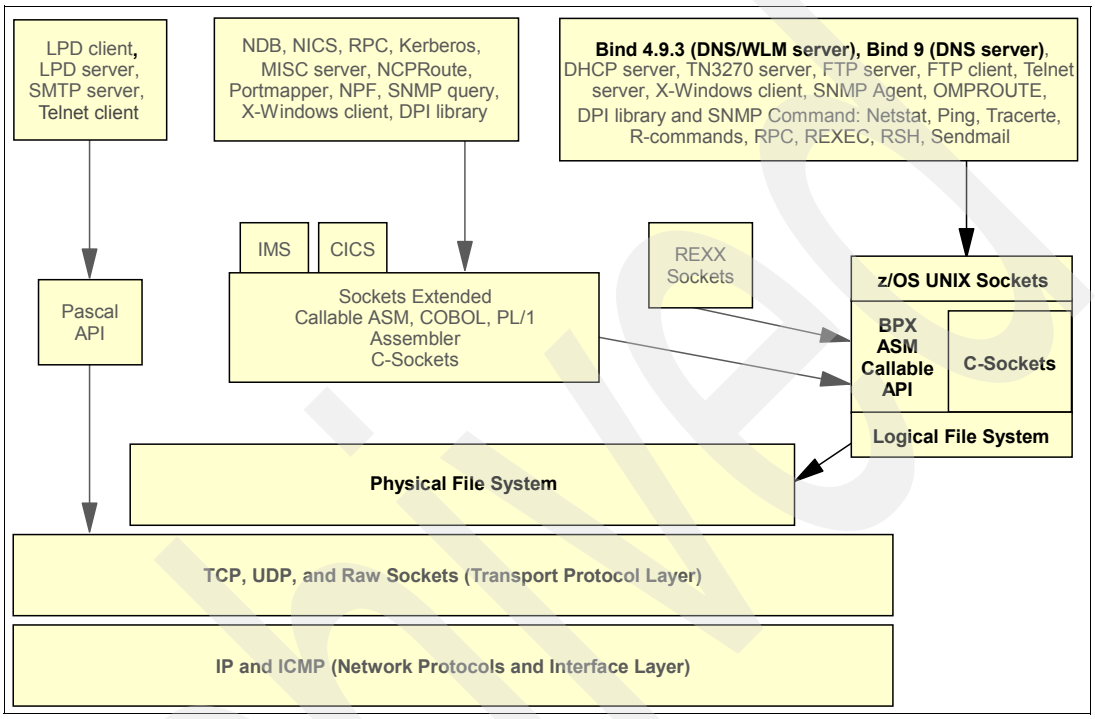


Figure 10-1 DNS service

Name servers provide names and IP addresses to clients called *resolvers*. The information helps to cross-reference an IP address to a name, or the name to an IP address.

This section discusses the following concepts:

- What is Domain Name Service?
- How does Domain Name Service work?
- How can Domain Name Service be applied?

10.1.1 What is Domain Name Service?

DNS was originally developed to make things easier for human users: to enable the use of symbolic names rather than the actual, numeric, 32-bit IPv4 or 128-bit IPv6 addresses upon which TCP/IP communications ultimately depends. The use of symbolic names supported by DNS, however, has turned out to be far more important than just a convenience because it provides separation of applications from their physical infrastructure. For example, without the use of symbolic names, if you needed to change the IP address of your TN3270E Telnet server (perhaps because you are moving the server) you would have to change the (potentially) thousands of clients configured to use its original IP address. With symbolic names, you need only change the DNS entry that corresponds to the TN3270E Telnet server name and the clients will learn of the changed IP address when they use its name.

DNS also provides IP address-to-host name mapping. The DNS defines a special domain called *in-addr.arpa* to translate IPv4 addresses to host names, and the *ip6.int* and *ip6.arpa*

domains for IPv6 address-to-host name translation. This kind of mapping is useful for producing output (host names) that is easy to read.

10.1.2 How does Domain Name Service work?

The relationships between resolvers (the clients) and DNS servers are illustrated in Figure 10-2. The authoritative servers are the Primary and Secondary servers with IP addresses of a.b.c.d and e.f.g.h. The DNS used with ADNR is an authoritative server.

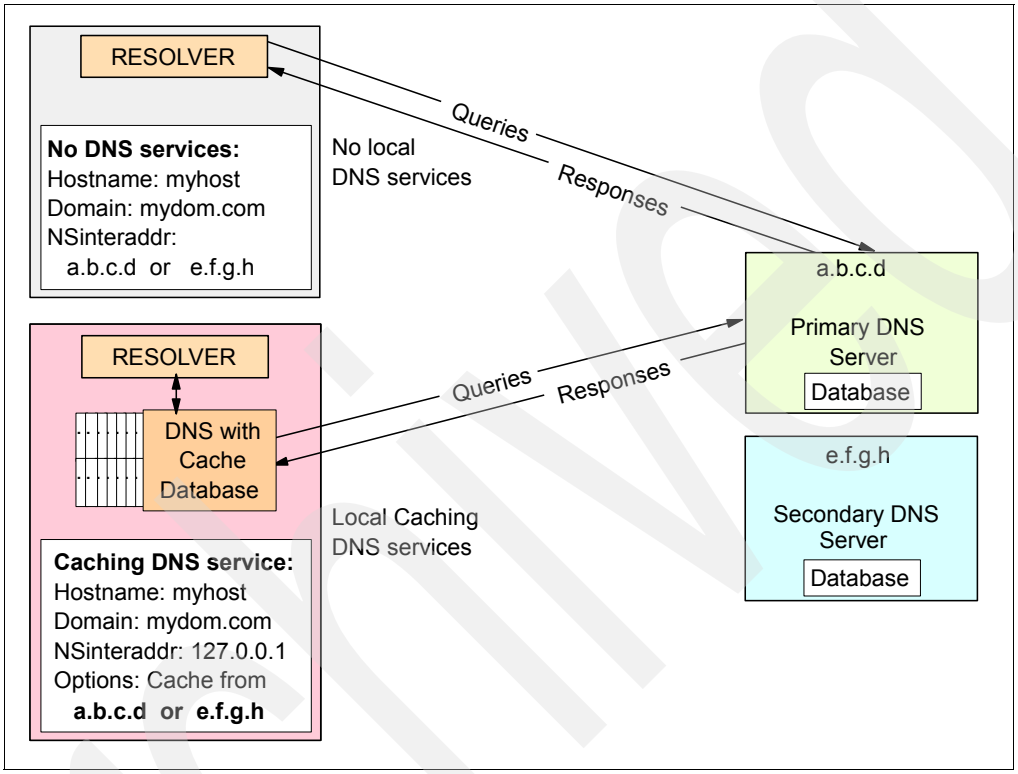


Figure 10-2 Caching DNS server and the resolver

DNS organizes the hosts in a network into *domains*. A domain is a group of hosts that share the same name space in the domain hierarchy and are usually controlled within the same organization. A special domain known as the *root* domain exists at the top of the hierarchy. The root domain servers store information about server hosts in the root domain and the name servers in the delegated, top-level domains, such as com (commercial), edu (education), and mil (military). The name servers in the top-level domain, in turn, store the names of name servers for their delegated domains, and so on.

The complete name of a host, also known as the fully qualified domain name (FQDN), is a series of labels separated by periods. Each label represents an increasingly higher domain level within a network. The complete name of a host connected to a large network generally has more than one subdomain, as shown in the following examples:

```
wtsc30.itso.ibm.com
wtsc31.itso.ibm.com
itsodns.itso.ibm.com
```

A domain name server requires the FQDN. The client resolver combines the host name with the domain name to create the FQDN before sending the name resolution request to the domain name server.

A zone consists of the resources within a single domain (for example, commercial or .com) or subdomain (for example, raleigh.ibm.com). Typically, a zone is administered by a single organization or individual. For DNS performance or availability reasons, you might want to partition your name space into zones and make each DNS server the *primary* for part of the name space and the *slave* (backup) for another part.

10.1.3 How can Domain Name Service be applied?

DNS can be implemented on the z/OS platform as one or more types of DNS servers. These types include authoritative, caching-only, forwarders, and stealth. A single server can perform multiple functions. For example, an authoritative DNS server can provide caching for names from other zones.

In this book we discuss these types of DNS services:

- Authoritative DNS server
- Caching-only DNS server
- Automated Domain Name Registration (ADNR)

Having your own authoritative DNS server gives you administrative control over your own space of names. Therefore, if you need a new name, you can simply add it to your server database rather than having to ask someone else to allocate a name for you. It also puts you in control of the availability and scalability of your own DNS service—not having to depend upon the DNS service provided by another organization.

You can use caching servers to create a cache of responses to frequently requested queries and reduce the number of queries made to master servers. The caching server stores data for a period of time determined by the time-to-live (ttl) value. Also, there is no clerical maintenance to be performed with a caching-only server.

Note: Configure your resolver so that it will check with the local caching-only DNS server first. However, if that server is unavailable for any reason, your resolver should also be able to direct queries to an external DNS (by using multiple NSINTERADDR statements in TCPIP.DATA. See Example 10-15 on page 293 for an example of this setup.

Most organizations have little need to design, implement, manage, and administer their own name space and authoritative DNS services for their IBM System z environment. Rather, that role is usually handled more globally for the organization and is usually implemented on external (non-z/OS) platforms. However, a caching-only DNS server on the mainframe may play an important role, especially to improve the performance of z/OS Web servers and WebSphere® applications. Caching DNS information on board can reduce CPU utilization and increase responsiveness to the numerous Web transactions that require DNS services.

We therefore focus our implementation on setting up and using a BIND 9 Caching-only DNS on the z/OS system.

Because we recommend a BIND 9 DNS, we need to discuss some BIND history and point out some compatibility issues between BIND 4.9.3 and BIND 9:

- BIND 9 allows IPv6-type records in zone data and better transaction security among servers and clients, as well as zone data authentication capability. It also has the rndc utility to replace and complement UNIX signals for name server local and remote control.
- The configuration file for the name server changed in both name and syntax between BIND 4.9.3 and BIND 9. The *dnsmigrate* tool aids in converting a BIND 4.9.3 named.boot file into a BIND 9 named.conf file. Through a statement of direction, IBM has indicated that the BIND 4.9.3 version of the name server is no longer being enhanced and will be

dropped in a future release. We strongly urge you to migrate any BIND 4.9.3 implementation you may have to BIND 9.

- The BIND 9.2 name server makes DNS server-to-server and client-to-server IPv6 connections possible, adding name server configuration options for IPv6 connections and tuning. BIND 9.2 also provides the rndc utility with a larger set of commands than was available with BIND 9.1. BIND 9.2 rndc is not compatible with BIND 9.1 rndc.
- The BIND 9 *nsupdate* utility enables client hosts and many DHCP servers to dynamically and securely register their name and address mappings. The z/OS BIND 9 name server is generally compatible with network DHCP servers. The z/OS DHCP server is compatible with the z/OS BIND 4.9.3 name server, but is incompatible with the BIND 9 name server.

Note: There are a number of considerations to be aware of when running the BIND 9 server in a multiple stack environment (CINET). Refer to the BIND 9 advanced topics section in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details about these restrictions.

10.2 Authoritative DNS server

An authoritative DNS server is responsible for one or more zones, managing the names and IP address associations within that zone. For reasons stated earlier in “How can Domain Name Service be applied?” on page 282, we provide only a description of the authoritative DNS server, and do not show its configuration nor implementation.

10.2.1 Description of an authoritative DNS server

Authoritative DNS servers are designated network nodes that maintain a database of information about all nodes in some part (zone) of the domain name space. Refer to Figure 10-2 on page 281 to see the role of the authoritative server contrasted with that of the caching-only server.

If a domain name server receives a query about a host for which it has information in its database or cache, it returns all the address records associated with the host to the client (some hosts might have more than one IP address). If the domain name server does not have the requested information, it can query other name servers for it. This process is called iterative resolution. The local name server successively queries other name servers, each of which responds by referring to a name server that is closer to the authoritative name server for the target domain. Ultimately, the local name server queries the authoritative name server and gets an answer. If the information about a requested host name does not exist or if a name server does not know where to go for the information, it sends a negative response back to the client.

Updates and maintenance changes are reflected in the master name server. The slave name servers update their databases by contacting the master name server at regular intervals or possibly (BIND 9) after being notified of an update by the master name server.

There are two types of authoritative servers: master (primary) and slave (secondary). Each zone must have only one master name server, and it should have at least one slave name server for backup to eliminate the single point of failure for this important service. Any given name server can take on either role, as defined by its configuration file.

Dependencies of an authoritative DNS server

If you set up your own authoritative DNS server you will, by definition, be required to manage and administer the space of names in your zone. In addition, given that yours is the *final*

authority about the names in your zone, you must be careful to implement sufficient availability and scalability for your server. If your authoritative DNS service becomes unavailable (due to overload or outage), even if the physical network infrastructure remains in tact, users will perceive an outage for any application accessed by name in your zone.

Note: If you use a VIPA address in the DNS for zone delegation, make sure that VIPA address is the SOURCEVIPA address on that stack. Otherwise, certain name server queries could fail. For further information, refer to 2.5.5.10 in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Considerations for using an authoritative DNS server

While having your own authoritative DNS server gives you a great deal of control, it comes with significant *responsibility*:

- As mentioned, you need to design your DNS service for sufficient scalability and availability to meet your business requirements. Unavailability of DNS service would likely result in significant application *outage*, as perceived by users.
- In addition, you need to provide ongoing management and administration of your DNS servers.
- Finally, providing DNS service could require significant processor resources, especially if you implementation requires security (DNSSEC, authenticating DNS data with digital signatures) implementation.

10.3 Caching-only DNS server

All name servers cache (store) the data they receive in response to a query. As the name suggests, caching-only DNS servers have no zone responsibility and must use other name servers to resolve name requests that they receive. They do, however, cache the resulting information and use it to answer future queries.

Note: When planning to set up caching-only DNS servers in your environment, check your current Resolver configuration and make sure the host names you plan to resolve by the caching-only DNS servers are not being resolved by the Resolver. The search order for the base Resolver configuration files is:

1. GLOBALTCPIPDATA
2. RESOLVER_CONFIG environment variable
3. /etc/resolv.conf
4. SYSTCP DD-name
5. userid.TCPIP.DATA
6. jobname.TCPIP.DATA
7. SYS1.TCPPARMS(TCPDATA)
8. DEFAULTTCPIPDATA
9. TCPIP.TCPIP.DATA

In this section we discuss the following topics:

- Description of a caching-only DNS server
- Configuration of a caching-only DNS server
- Activation and verification of a caching-only DNS server

10.3.1 Description of a caching-only DNS server

A caching-only server is not authoritative for any domain. When a caching-only server receives a query, it checks its cache for the requested information. If it does not have the information, it queries a local name server or a root name server, passes the information to the client, and caches the answer for future queries. The names and addresses of the root name servers are acquired from the servers listed in the hints file and the name and file path of the hints file are specified in the name server's configuration file. Refer to Figure 10-2 on page 281 to see the role of the authoritative server contrasted with that of the caching-only server.

Dependencies of a caching-only DNS server

Because the caching-only DNS server has no domain name database of its own to manage, it is completely dependent upon other DNS servers.

If you want to use an application-specific Dynamic VIPA (DVIPA) as the address of the name server, then the name server must BIND to the well-known port for DNS (UDP port 53) on the DVIPA.

Considerations for using a caching-only DNS server

All cached information is lost if a caching-only DNS server is restarted. Also, by definition, a caching-only DNS server gives your no control over your name space. Rather, you will need to work with your local naming authority (the person or people who control the authoritative name server for your zone).

Note: Use z/OS UNIX or TSO nslookup with each NSINTERADDR used in TCPIP.DATA to ensure you receive the expected results. Some name server clients on other platforms may require the address you specify for the name server to match the source IP address in the response from the name server.

For example, if a static VIPA address is specified as the address of the name server, and IPCONFIG SOURCEVIPA is not specified in PROFILE.TCPIP, then nslookup on some platforms will discard the returned packet because it will have the destination address of the physical interface instead of the VIPA interface.

10.3.2 Configuration of a caching-only DNS server

The name resolution process is an example of a client/server relationship in which clients, through their resolvers, request name resolution service from a name server.

z/OS Communications Server: IP Configuration Guide, SC31-8775, contains a comprehensive discussion on setting up the BIND 9 Caching-only server. Refer to that manual for details. A summarized checklist is provided here to assist with understanding the above reference.

The tasks for configuring a caching-only server are discussed in the following sections:

- Create a BIND 9 configuration file for the server.
- Create an environment variables file for the server.
- Update the TCP/IP stack profile port reservation list for the server.
- Create the name server start procedure.
- Create the hints file (root server).
- Create the loopback file.
- Configure logging options for the server.
- Implement the syslog daemon (syslogd) or alternative log file.

- Determine the run mode of the server: swappable or nonswappable.

Create a BIND 9 configuration file for the server

We created a working directory (dnsdatb31) in the z/OS UNIX file under the main /etc directory where all of our DNS server files could be placed. We explicitly defined some of the files. The others were dynamically created by the server task. We coded a directory option statement pointing to the desired directory. A directory list of /etc/dnsdatb31 resulted in the display shown in Example 10-1.

Example 10-1 Listing of the directory for the NAMED9B server, /etc/dnsdatb31

```
Directory List
EUID=0   /SC31/etc/dnsdatb31/
  Type  Perm  Permission  Changed-EST5EDT   Owner      Filename    Row 1 of 8
_ File   644   rw-r--r--  2007-10-02 18:47  BPXROOT    named9b.pid
_ File   666   rw-rw-rw-  2007-10-02 18:47  BPXROOT    log.log
_ Dir    755   rwxr-xr-x  2007-09-30 18:20             ..
_ File   777   rwxrwxrwx  2007-10-02 18:11  CS07       named9b.env
_ File   777   rwxrwxrwx  2007-10-02 18:32  BPXROOT    named9b.conf
_ File   777   rwxrwxrwx  2007-10-02 18:46  CS07       db.loopback.v9
_ File   777   rwxrwxrwx  2007-09-11 16:13  CS07       db.cache
_ Dir    777   rwxrwxrwx  2007-10-02 19:23  CS07       .
```

Example 10-2 is the configuration file for a caching-only server.

Example 10-2 Configuration file for the NAMED9B task

```
BROWSE -- /SC31/etc/dnsdatb31/named9b.conf --------- Line 00000000 Col 001 050
#  named.conf file for named9b for sc31

options {
   directory "/etc/dnsdatb31";
   pid-file "/etc/dnsdatb31/named9b.pid";
   statistics-file "/etc/dnsdatb31/named9b.stats";
   forward only;
   forwarders {10.12.6.7; 10.12.12.7; 10.12.4.7;};
   max-ncache-ttl 1800;
   max-cache-ttl 36000;
};

logging {
   channel "myfile" {
      file "/etc/dnsdatb31/log.log"
       versions 99
       size 30m ;
      severity info ;
      };
   category "default" { "myfile"; };
   category "queries" { "myfile"; };
   category "general" { "myfile"; };
   category "config"  { "myfile"; };

zone "0.0.127.in-addr.arpa" in {
   type master;
   file "/etc/dnsdatb31/db.loopback.v9";
};

zone "." in {
   type hint;
   file "/etc/dnsdatb31/db.cache";
```

```
};
```

A sample configuration file for a caching server is located in /usr/lpp/tcpip/samples/caching.conf, and can be copied to the file that you want to use for your configuration file. Because we chose to put all of our files in /etc/dnsdatb31, we copied the sample file to /etc/dnsdatb30/named9b.conf, and pointed our started task proc JCL to it by using the -c parameter.

Create an environment variables file for the server

We created an environment variables file to specify stack affinity, resolver config, job name, and DNS version. The BIND 9 server is a generic server and binds to all interfaces on all stacks in a CINET environment. If you want it to have stack affinity, you can specify stack affinity in the environment variables file like we did. The server can be configured to listen on a subset of available interface addresses if desired. It is not necessary to specify the job name, because the system suffixes the started task name with the number one (1) when BIND 9 starts. Our started task was NAMED9B, but specified a job name of NAMEDDD, and the forked job resulted in a name of NAMEDDD1. If you want the job name to be different from the started task name, you can set the job name in the environment variables file. The STDENV file that we used is shown in Example 10-3.

Example 10-3 Sample STDENV file

```
BROWSE -- /SC31/etc/dnsdatb31/named9b.env ---------- Line 00000000 Col 001 045
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
_BPX_JOBNAME=NAMEDDD
DNS_VERSION=v9
```

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

RECFM=F must not be used, because it allows padding of the record with blanks after the environment variable value. When the variable represents a file name, the padded value could cause a file-not-found condition because the padded blanks are considered part of the name of the file in UNIX System Services. If the standard environment file is in MVS and is not allocated with RECFM=V, the results can be unpredictable.

Update the TCP/IP stack profile port reservation list for the server

The PORT reservation list would look like Example 10-4.

Example 10-4 Port reservation for the DNS server

```
PORT
   53 TCP NAMEDDD1
   53 UDP NAMEDDD1
```

The PORT reservation for port 53 can specify the started task name as the job name (suffixed with the number one (1)), or the UNIX shell address space name can be specified. Because it is less clerical effort, you can use OMVS to avoid any name conflicts in the future, especially if you change the name of the started task.

```
PORT
   53 TCP OMVS
   53 UDP OMVS
```

Create the name server start procedure

A sample procedure for the BIND 9 server is located in SEZAINST(NAMED9). We modified the sample to specify our own configuration file and environment variable file. The start procedure JCL for the BIND 9 DNS server is stored in a system proclib. Make sure you use a valid PARM string acceptable to BPXBATCH. It requires that the PARM string field be a contiguous string to column 71, with a continuation character in column72, then the remainder of the parm field continuing on the next line in column 16. Also notice the specification of the configuration file in the PARM field, using the -c parameter. The environment variables file is pointed to by using the //STDENV DD statement. BPXBATCH automatically detects the presence of the //STDENV DD statement, and you do not have to use the _CEE_ENVFILE environment variable to point to it. The debug trace level can be specified when starting the procedure. The default debug level is set to zero. Additional parameters can be included in the PARMS= specification at startup, as seen in Example 10-5.

Example 10-5 JCL procedure for the BIND 9 DNS server

```
BROWSE    SYS1.PROCLIB(NAMED9B) - 01.27             Line 00000000 Col 001 080
//NAMED9B  PROC PARMS='-d 0'
//NAMED9B  EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
// PARM=('PGM /usr/lpp/tcpip/sbin/named -V v9 -c /etc/dnsdatb&SYSCLONE.X
//             /named9b.conf &PARMS')
//STDENV   DD PATH='/etc/dnsdatb&SYSCLONE./named9b.env',
//             PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Create the hints file (root server)

The hints file does not contain cached data, nor does the name server provide other hosts with the information contained in the hints file. A *forward-only* server is the only type of name server that does not require a hints file. If your BIND 9 server is going to run on a system that is behind a firewall, then your hints file will point to one or more of your normal DNS servers that can resolve intranet and Internet names. However, if your system is not behind a firewall and has access to the Internet DNS root servers, then you must install the hints file provided by the Internic. To obtain a hints file, point your Web browser at `ftp://ftp.rs.internic.net` and retrieve the file named.root from the domain subdirectory.

Because we are defining a caching-only server, we chose to use forward-only, specifying one or more of our own internal DNS servers, and leave the hints file (/etc/dnsdatb31/db.cache) empty.

> **Note:** Before you specify the remote DNS servers IP addresses in *forwarders* statement with *forward only*, make sure your Caching-only server can access them, otherwise the local caching-only server could not resolve the host names.

Create the loopback file

The sample loopback file for BIND 9 is located in /usr/lpp/tcpip/samples/db.loopback.v9, and can be copied to the file you want to use for your loopback file. Because we placed all of our BIND 9 DNS related files into /etc/dnsdatb31, we copied the sample file to /etc/dnsdatb31/db.loopback.v9 and customized it to reflect our domain name. For a comprehensive discussion on the loopback file for caching servers, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Example 10-6 is the loopback file.

Example 10-6 Loopback file for BIND 9 DNS server

```
BROWSE -- /SC31/etc/dnsdatb31/db.loopback.v9 ------- Line 00000000 Col 001 065
;
;   /etc/dnsdata/db.loopback.v9
;
; Default TTL value, $TTL is for BAND 9
$TTL 86400
0.0.127.in-addr.arpa. IN SOA itsodns.itso.ibm.com. itsodns_adm.itso.ibm.com. ( 1
     1
     10800
     3600
     604800
     86400   )
0.0.127.in-addr.arpa.   IN   NS  itsodns.itso.ibm.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
```

1 The loopback file is a z/OS UNIX file, and the record length has no 72 character limitation on one line. The SOA statement should be followed by the authoritative_name_server, resp_person and the first parenthesis in the same line. Do not omit either of these, or you will receive the error messages shown in Example 10-7 in the UNIX log file of the name server (/etc/dnsdatb31/log.log) when the it starts up and loads the loopback file.

Example 10-7 Error messages for loopback file

```
EZZ9190I dns_rdata_fromtext: /etc/dnsdatb31/db.loopback.v9:17: near eol: unexpected end of
input
EZZ9688I zone 0.0.127.in-addr.arpa/IN: loading master file /etc/dnsdatb31/db.loopback.v9:
unexpected end of input
```

For the statement syntax in loopback file, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Configure logging options for the server

A wide variety of logging options for the name server can be configured via the *logging* statement. Its *channel* phrase associates output methods, format options, and severity levels with a name that can then be used with the category phrase to select how various classes of messages are logged. Only one logging statement is used to define as many channels and categories as are wanted. For details of the logging statement, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for a discussion about how to use logging. For details on the syntax and format of the logging statement, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776. Example 10-8 shows the statements we used. Our statements are not typical, but simply show the types of logging channels and categories you can create and manage.

Example 10-8 Sample logging options

```
logging {
   channel "myfile" {
      file "/etc/dnsdatb31/log.log"
       versions 99
       size 30m ;
      severity info ;
      };
   category "default" { "myfile"; };
   category "queries" { "myfile"; };
   category "general" { "myfile"; };
```

```
   category "config"  { "myfile"; };
};
```

Implement the syslog daemon (syslogd) or alternative log file

For BIND 9 only, unless syslogd is running, no messages will be produced by NAMED during name server initialization. This includes event logging and any syntax errors that might be detected in the configuration file. Not performing this step complicates problem determination, especially for failures at startup. For descriptions of the syslog file and the syslog daemon, see Chapter 1, “Syslogd” on page 1.

Determine the run mode of the server: swappable or nonswappable

If you want to run the name server as *swappable*, you must have the BPX.STOR.SWAP FACILITY class profile defined to RACF with no universal access. To do this, enter the following commands from a RACF user ID:

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
SETROPTS RACLIST(FACILITY) REFRESH
```

If you want the name server to run in a *nonswappable* state, use the following commands:

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
PERMIT BPX.STOR.SWAP CLASS(FACILITY) ID(userid) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

The *userid* in the above command is the user ID under which your BIND 9 started task is intended to run.

10.3.3 Activation and verification of a caching-only DNS server

The startup of the server and its responsiveness can be verified as follows:

- Start the name server task
- Verify that the name server started correctly.
- Use netstat to verify that the server is listening on the intended port.
- Verify that the name server can accept queries.
- Browse the log to see the latest query activity just created.
- Stop the caching-only DNS server to verify it will shut down cleanly.

Start the name server task

The BIND 9 server can be started using the MVS START (S) command, the z.OS UNIX shell named command, or via automations. We started our NAMED9B procedure using the MVS START command:

```
S NAMED9B
```

Sample messages issued at startup are shown in Example 10-9. The original job spawns a new job called NAMED9B1, and then the original job terminates.

Example 10-9 NAMED9 startup messages

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 3 0  --  N O D E
S NAMED9B
$HASP100 NAMED9B  ON STCINRDR
IEF695I START NAMED9B  WITH JOBNAME NAMED9B  IS ASSIGNED TO USER TCPIP
   , GROUP TCPGRP
$HASP373 NAMED9B  STARTED
$HASP395 NAMED9B  ENDED
+EZZ9095I STARTING NAMED, BIND 9.2.0
```

```
IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=0094.
+EZZ9130I NAMED, BIND 9.2.0 IS RUNNING
```

After startup, the process ID of the NAMED9B task is placed into the PID file, as shown in Example 10-10.

Example 10-10 PID file containing the process ID of the NAMED9B task

```
BROWSE -- /SC31/etc/dnsdatb31/named9b.pid ---------- Line 00000000 Col 001 005
65692
```

The PID value can be used on a **kill** command to stop the task. If the NAMED9B task was started using the MVS START (S) command, the MVS STOP (P) command can be used to stop the task, as in Example 10-11.

Example 10-11 Stopping the NAMED9B task using the UNIX shell

```
kill -TERM 65692
or
kill -TERM $(cat /etc/dnsdatb31/named9b.pid)
or
P NAMED9B
```

Verify that the name server started correctly

Browse the log where you directed the server to log and verify the results of startup. As shown in Example 10-12.

Example 10-12 NAMED9 log immediately after startup

```
BROWSE -- /SC31/etc/dnsdatb31/log.log -------------- Line 00000000 Col 035 114
EZZ9688I zone 0.0.127.in-addr.arpa/IN: loaded serial 1
```

There could be warning messages indicating that optional files were not found. Processing continues normally without them. However, if you intend to use **rndc** to manage your server, then you must create the rndc environment (refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781).

Use netstat to verify that the server is listening on the intended port

Netstat shows current socket and UDP connections. Verify that the name server that you started is listening on the intended stacks, interfaces, and port. The resulting netstat display that we used is shown in Example 10-13.

Example 10-13 Verifying NAMED9B connections with NETSTAT

```
netstat conn
.
NAMEDDD2 0000C642 Listen
  Local Socket:   10.1.1.20..53
  Foreign Socket: 0.0.0.0..0
NAMEDDD2 0000C640 Listen
  Local Socket:   10.1.1.21..53
  Foreign Socket: 0.0.0.0..0
NAMEDDD2 0000C644 Listen
  Local Socket:   127.0.0.1..53
  Foreign Socket: 0.0.0.0..0
.
NAMEDDD2 0000C645 UDP
```

```
  Local Socket:   0.0.0.0..14619
  Foreign Socket: *..*
NAMEDDD2 0000C646 UDP
  Local Socket:   ::..14620 (IPV6_ONLY)
  Foreign Socket: *..*
NAMEDDD2 0000C641 UDP
  Local Socket:   10.1.1.20..53
  Foreign Socket: *..*
NAMEDDD2 0000C643 UDP
  Local Socket:   127.0.0.1..53
  Foreign Socket: *..*
NAMEDDD2 0000C63F UDP
  Local Socket:   10.1.1.21..53
  Foreign Socket: *..*
```

Verify that the name server can accept queries

First, use the normal TSO *nslookup* process to verify that you can retrieve DNS information using the usual TCPDATA file that has the standard *external DNS server* specified on the NSINTERADDR/NAMESERVER statement. This method used file 'TCPIPB.TCPPARMS(*DATAB31*)' and is shown in the following example. It does not make use of the new caching-only server that we just started. Pointing to and using an external DNS is shown in Example 10-14.

Example 10-14 Uses external DNS server

```
BROWSE TCPIPB.TCPPARMS(DATAB31) - 01.05            Line 00000000 Col 001 080
TCPIPJOBNAME TCPIPB
HOSTNAME WTSC31
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  10.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

free ddn(systcpd)
alloc ddn(systcpd) dsn('TCPIPB.TCPPARMS(DATAB31)') shr
nslookup

 EZB3170I Default Server:  itsodns.itso.ibm.com
 EZB3172I Address:  10.12.6.7
 EZB3042I >
wtsc30
 EZB3170I Server:  itsodns.itso.ibm.com
 EZB3172I Address:  10.12.6.7
 EZB3170I Name:    wtsc30.ITSO.IBM.COM
 EZB3172I Address:  10.1.1.10

 EZB3042I >
wtsc31
 EZB3170I Server:  itsodns.itso.ibm.com
 EZB3172I Address:  10.12.6.7
 EZB3170I Name:    wtsc31.ITSO.IBM.COM
 EZB3172I Address:  10.1.1.20
 EZB3042I >
```

Next, use the TSO nslookup process to verify that you can retrieve DNS information using the special TCPDATA file that has been set up with the NSINTERADDR/NAMESERVER statement pointing to the *loopback address* (127.0.0.1) instead of pointing to the remote name server directly. An example of this method uses file 'TCPIPB.TCPPARMS(*DATAB31N*). It indicates to the resolver to use our own DNS server that is listening on port 53 on our own loopback address, which is the caching-only server that we just started. Pointing to and using the local server is shown in Example 10-15.

Example 10-15 Local caching DNS specified, loopback address

```
BROWSE    TCPIPB.TCPPARMS(DATAB31N) - 01.02          Line 00000000 Col 001 080
TCPIPJOBNAME TCPIPB
HOSTNAME WTSC31
DOMAINORIGIN  ITSO.IBM.COM
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  127.0.0.1   <=== Was 10.12.6.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1

alloc ddn(systcpd) dsn('tcpipb.tcpparms(dat30tst)') shr
nslookup

 EZB3170I Default Server:  localhost
 EZB3172I Address:  127.0.0.1
 EZB3042I >
wtsc30
 EZB3170I Server:  localhost
 EZB3172I Address:  127.0.0.1

 EZB3110I Non-authoritative answer:
 EZB3170I Name:    wtsc30.ITSO.IBM.COM
 EZB3172I Address:  10.12.4.211

 EZB3042I >
wtsc31
 EZB3170I Server:  localhost
 EZB3172I Address:  127.0.0.1

 EZB3110I Non-authoritative answer:
 EZB3170I Name:    wtsc31.ITSO.IBM.COM
 EZB3172I Address:  10.12.4.213

 EZB3042I >
```

Compare the server names and addresses between the two methods. Notice the first method used server *itsodns.itso.ibm.com* at *10.12.6.7*, but the second method used our *localhost* at *127.0.0.1*. Also note that the caching-only server issued message `EZB3110I Non-authoritative answer`.

Next, use TSO DIG to query for the same names that were used for nslookup. Ours names were *wtsc30* and *wtsc31*. This is shown in Example 10-16.

Example 10-16 DIG using loopback TCPDATA file, pointing to local loopback caching DNS

```
dig wtsc30

EZB3316I ;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 3
```

```
EZB3328I ;; Ques: 1,Ans: 1,Auth: 0,Addit: 3
EZB3332I ;; QUESTIONS:
;;      wtsc30.ITSO.IBM.COM, type = A, class = IN

EZB3359I ;; ANSWERS:
wtsc30.ITSO.IBM.COM.    A       10.1.1.10

dig wtsc31

EZB3316I ;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 3
EZB3328I ;; Ques: 1,Ans: 1,Auth: 0,Addit: 3
EZB3332I ;; QUESTIONS:
;;      wtsc31.ITSO.IBM.COM, type = A, class = IN

EZB3359I ;; ANSWERS:
wtsc31.ITSO.IBM.COM.    A       10.1.1.20
```

Browse the log to see the latest query activity just created

Checking the log once again, as in Example 10-17, shows the results of the query activity (nslookup and dig) that was just performed.

Example 10-17 NAMED9 Log after using nslookup and dig for WTSC30 and WTSC31

```
BROWSE -- /SC31/etc/dnsdatb31/log.log -------------- Line 00000000 Col 035 114

EZZ8828I client 127.0.0.1#1260: query: 1.0.0.127.in-addr.arpa IN PTR
EZZ8828I client 127.0.0.1#1261: query: wtsc30.ITSO.IBM.COM IN A
EZZ8828I client 127.0.0.1#1262: query: wtsc31.ITSO.IBM.COM IN A

EZZ8828I client 127.0.0.1#1289: query: . IN A
EZZ8828I client 127.0.0.1#1290: query: wtsc30.ITSO.IBM.COM IN A
EZZ8828I client 127.0.0.1#1296: query: wtsc31.ITSO.IBM.COM IN A
```

Stop the caching-only DNS server to verify it will shut down cleanly

When you stop the BIND 9 server using the MVS STOP (P) command, it logs shutdown messages, as shown in Example 10-18.

Example 10-18 NAMED9 log immediately after shutdown of NAMED9 task

```
BROWSE -- /SC31/etc/dnsdatb31/log.log -------------- Line 00000000 Col 035 114

 1327ec20: EZZ9131I shutting down
 1327ec20: EZZ9045I no longer listening on 10.1.1.10#53
 1327ec20: EZZ9045I no longer listening on 10.1.1.20#53
 1327ec20: EZZ9045I no longer listening on 127.0.0.1#53
```

10.4 Automated domain name registration

The automated domain name registration (ADNR) application enables you to dynamically add or delete application-specific hostnames (and the addresses of those applications) in name servers according to application availability. The DNS names managed by ADNR can be names that represent a specific instance of an application within the sysplex, names that represent individual systems within the sysplex, as well as names that represent the entire sysplex.

This section includes the following topics:

- Description of ADNR
- Configuration of ADNR
- Activation and verification of ADNR

10.4.1 Description of ADNR

The name server or name servers that ADNR updates can be z/OS BIND 9 name servers, or non-z/OS name servers that support BIND 9 dynamic update. ADNR supports both IPv4 and IPv6 addresses. Figure 10-3 depicts the communication relationship between ADNR, the Load Balancing Advisor (LBA), and the LB Agents.

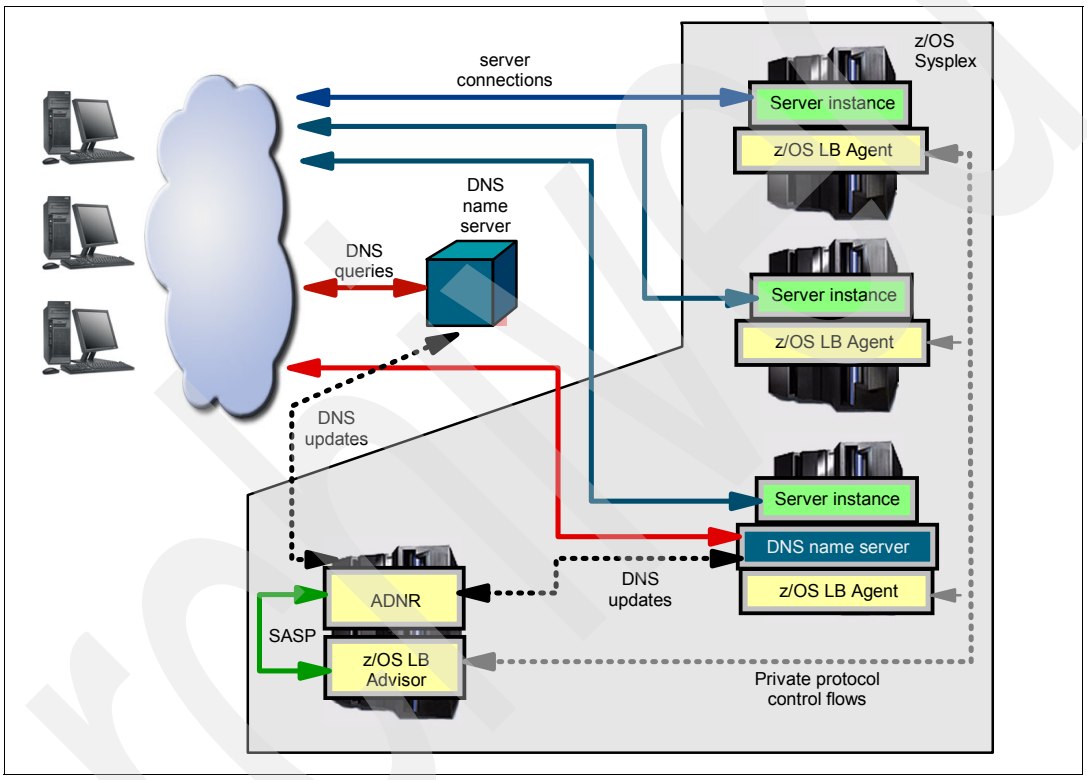


Figure 10-3 Diagram representing the flow among ADNR, Load Balancing Advisor, and agents

ADNR is configured with information about sysplex resources and their assigned DNS names. These resources are configured on a group basis. ADNR registers its configured information about sysplex resources with a z/OS Load Balancing Advisor application. The Advisor disseminates this information to the z/OS Load Balancing Agents, which report back to the Advisor about the availability of the resources registered by ADNR. The Advisor subsequently reports to ADNR any changes in the availability of those resources.

For each group of resources that the Advisor reports as available, ADNR adds a DNS name to the name server that represents the entire group of resources in that group, and maps that name to the IP addresses of the available resources in that group. The interaction between ADNR and the Load Balancing Advisor is through the SASP protocol (Server/Application State Protocol).

10.4.2 Configuration of ADNR

Note: You do not need to have Sysplex Distributor implemented in your environment to have ADNR configured.

For performance purposes, it is advantageous to run ADNR and LBA Advisor in the same image. The DNS server can be in the same image, too. Perform these tasks to configure ADNR:

- Create the ADNR configuration file
- Configure your TCPIP.DATA to point to the DNS server
- Configure the z/OS BIND9 DNS server file
- Create the LB Advisor configuration file
- Create the LB Agent configuration files for sysplex LPARs
- Define security server profiles for ADNR

Create the ADNR configuration file

In Example 10-19 we note some basics elements of the ADNR configuration file.

Example 10-19 ADNR configuration file example

```
debug_level             128            # Error, Warning, Event, Info
uuid                    IBM_sysplex_adnr
dns                     dnsserv        # Label used by other stmts
                                       #  and commands
{
  dns_id                10.1.2.99..53  # Linux name server

# This zone will contain all addresses which may be used by intranet
# clients to reach applications within the sysplex.

  zone                  myzone
                                       # Label used by other stmts
                                       #  and commands
  {
    domain_suffix       example.com
                                       # Zone name in name server
#   update_key          mvsplex_update_key
                                       # Key to sign dyn. updates
#   transfer_key        mvsplex_transfer_key
                                       # Key for zone transfers
    ttl                 60             # Time To Live for zone
  } # end of zone
} # end of dns

#-------------------------------
gwm                     z/os_lba_advisor
{
  gwm_id               10.1.9.30..3860 # LBA lb_connection_v4 address
  host_connection_addr  10.1.1.30      # Local address
} # end of gwm
# --------------------------ok--------------------------------
 host_group              itso_sysplex                          1

  {
  host_group_name       SC3Xgroup                             2
  dns                   dnsserv
  zone                 example.com
```

```
  member                SC31                                          3
  {
    host_name           WTSC31        # Prepended to domain suffix  4
    ipaddrlist          sc31_vipa_addrs                               5
  }
  } # end of host_group definition
ipaddrlist              sc31_vipa_addrs                               5
  {
    ipaddr                10.1.1.20       # static VIPA on sc31       6
  } # end of ipaddrlist
```

- 1 *Host_group* statement represents the mapping of host name-to-IP address in the DNS server.
- 2 *Host_group_name* is the name of the group of hosts to be updated in the name server.
- 3 The statement *member* defines a member for a given group.
- 4 You should define the name of the individual host to be updated in the name server through the *host_name* parameter.
- 5 The *ipaddrlist* parameter is associated with the member host_name.
- 6 The *ipaddr* parameter is the static VIPA address representing the stack.

In our configuration we used a basic customization, including just one IP address to monitor, as shown in Example 10-19 on page 296. When you define a member in the ADNR configuration, you are saying that this member should be monitored by the Agents and if this address does not respond the Agent will inform the Advisor. The Advisor then informs ADNR so it can proceed with an update into the DNS server. Our configuration is not using authentication, so the statements *key*, *transfer-key*, and *update_key* are not used. Use of these keys causes the data sent by ADNR to be signed by ADNR and authenticated by the name server. If you need the authentication function, you can get more informations about how to configure it in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Configure your TCPIP.DATA to point to the DNS server

Example 10-20 shows the TCPIP.DATA file.

Example 10-20 TCPIP.DATA file configuration

```
TCPIPJOBNAME TCPIPB
HOSTNAME WTSC30B
DOMAINORIGIN  EXAMPLE.COM  1
DATASETPREFIX TCPIPB
MESSAGECASE MIXED
NSINTERADDR  10.1.2.99    2
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
```

1 The domain should match the zone defined on the DNS server.

2 The Name Server IP address must be the IP address of the DNS server.

Configure the z/OS BIND9 DNS server file

Example 10-21 shows our DNS server configuration file (named.conf).

Example 10-21 DNS server config file for example.com

```
[root@dnsserv:etc] # more named.conf
// named.conf for example.com
logging {
     channel dnslog {
          file "/var/named/bind9.log" versions 3 size 3m;
          severity debug 10;
          print-time yes;
          print-severity yes;
          print-category yes;
          };
     category default {
          dnslog;
          };
     };
options {
     directory "/var/named";
     dump-file "/var/named/data/cache_dump.db";
     statistics-file "/var/named/data/named_stats.txt";
     version "9.2.4";
};
zone "example.com" in {
     type master;                                   1
     file "example.com.zone";                       3
     allow-update { any; };                         2
     };
```

1 The name servers that ADNR manages require a one-time setup. These name servers must be configured as the primary master name servers for the zones managed by ADNR.

2 We configured our zone statements with "allow-update { any; };", because the DNS server was in our test environment. This however, is an unsafe practice in a production environment. It allows the zone to be updated by unauthorized personnel or programs and could present a security risk.

3 Zones managed by ADNR must be updated exclusively by ADNR. The zones and zone files should not be edited by hand nor should anyone or any other program perform dynamic updates to the zones.

Create the LB Advisor configuration file

You must configure the Load Balancing Advisor, as shown in Example 10-22. An external load balancer is not required.

Example 10-22 Load Balancing Advisor configuration file

```
#Load Balancing Advisor Configuration
debug_level               7
update_interval           60
agent_connection_port     8100

agent_id_list
{
  10.1.1.20..8000
  10.1.1.10..8000
}
lb_connection_v4          10.1.9.30..3860
lb_id_list
{
```

```
  10.1.1.30
}
wlm serverwlm
```

Create the LB Agent configuration files for sysplex LPARs

Example 10-23 shows the LB Agent configuration files for SC30.

Example 10-23 Load Balancing Agent configuration file for SC30

```
debug_level              7
advisor_id               10.1.1.30..8100
host_connection          10.1.1.10..8000
```

Example 10-24 shows the LB Agent configuration file for SC31.

Example 10-24 Load Balancing Agent configuration file for SC31

```
debug_level              7
advisor_id               10.1.1.30..8100
host_connection          10.1.1.20..8000
```

Define security server profiles for ADNR

Copy the ADNR start procedure from SEZAINST(EZBADNRS) and customize it, as shown in Example 10-25.

Example 10-25 Start procedure example for ADNR

```
BROWSE    SYS1.PROCLIB(ADNR) - 01.05
//ADNR     PROC
//ADNR   EXEC PGM=EZBADNR,REGION=OK,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)/',
//            'ENVAR("_CEE_ENVFILE=DD:STDENV")/')
//CONFIG   DD DSN=TCPIPC.TCPPARMS(ADNRCNFC),DISP=SHR
//STDENV   DD DISP=SHR,DSN=TCPIP.SC&SYSCLONE..STDENV(ADNR&SYSCLONE.)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//CEESNAP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSMDUMP DD DUMMY
```

Define RACF USERID, STARTED, OPERCMD profiles for ADNR start procedure, and grant authority to start ADNR as follows:

```
ADDUSER ADNR DFLTGRP(OMVSGRP) OMVS(UID(9999) HOME('/tmp') PROGRAM('/bin/sh'))
RDEFINE STARTED ADNR.* STDATA(USER(ADNR))
SETROPTS CLASSACT(OPERCMDS)
SETROPTS RACLIST (OPERCMDS)
RDEFINE OPERCMDS (MVS.SERVMGR.ADNR) UACC(NONE)
PERMIT MVS.SERVMGR.ADNR CLASS(OPERCMDS) ACCESS(CONTROL) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

|ADNR must have read and write access to the directory specified on the HOME keyword. This directory becomes ADNR's *working directory*. ADNR creates and deletes temporary files in this directory during its operation. The UID value, nn, can be zero or nonzero. And the program specified on the user ID assigned to run ADNR must be */bin/sh*.

Note: ADNR is a multi-threaded application. If you define an unusually large number of name servers or zones to ADNR, you should check to determine whether the maximum number of threads allowed per process, represented by the THREADSMAX value in BPXPRMxx, is going to be exceeded. The number of threads required for ADNR is determined in the following way: (number of dns statements) + (number of zone keywords within all dns statements) + 3. You can customize the maximum number of threads allowed for ADNR by specifying the THREADSMAX keyword on the ADDUSER command.

10.4.3 Activation and verification of ADNR

After ADNR registers its resources with the Advisor, it waits for a period of time to ensure it has received all information from the Advisor about the availability of these resources. The *convergence interval* is normally two times the Advisor update interval. ADNR issues an event message when the convergence is complete.

To determine whether convergence is complete, display ADNR's GWM state as shown in Example 10-26, using the command **MODIFY procname,DISP,GWM**.

Example 10-26 Example showing the convergence between ADNR and Load Balancing Advisor

```
F ADNR,DISPLAY,GWM
EZD1254I GWM SUMMARY 663
GWM LABEL        : Z/OS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
1 OF 1 RECORDS DISPLAYED
```

Once the convergence completes, ADNR attempts to update all the name server's zones with the appropriate resource records. Records for resources that are no longer available are update-deleted from each zone and those that are available are update-added to each zone if they are not already present.

After this has occurred, the zone is in SYNCHRONIZED state and you can check it using the command **MODIFY procname,DISP,DNS,ZONES**.

ADNR examines the contents of zones specified on the *dns* statement during zone resynchronization.

We used the following steps to verify that when the TCP/IP in LPAR SC31 becomes unavailable its name (sc31.example.com) is removed from the DNS server. After restarting the TCPIP in that LPAR the name should be registered again on the DNS server. The steps we used are:

- Verify you can ping the sysplex member DNS name
- Stop the TCP/IP stack in the sysplex member
- Verify that the DNS status has changed for ADNR
- Verify that you cannot ping the sysplex member DNS name
- Re-start the TCP/IP stack and display the zones again
- Verify you can ping the sysplex member DNS name again

Verify you can ping the sysplex member DNS name

We issued the following command from SC32:

```
ping WTSC31.example.com
```

and received the output shown in Example 10-27.

Example 10-27 Ping from the SC32 system against the name registered on DNS server through ADNR

```
Pinging host WTSC31.EXAMPLE.COM (10.1.1.20)
Ping #1 response took 0.000 seconds.
```

Stop the TCP/IP stack in the sysplex member

Stop the TCP/IP stack on the sysplex member (in our case: SC31). This action will cause its IP address to become unavailable.

Verify that the DNS status has changed for ADNR

Issue the following console command:

```
F ADNR,DISP,DNS,ZONES,DETAIL
```

The display shows that the label of the sysplex member (in our case: SC31) is removing from the DNS server (see Example 10-28).

Example 10-28 Display of the zones on DNS when removing the TCP/IP stack

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 915
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/03/07 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : UPDATE-DELETE_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/03/07 19:05:47
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : UPDATE-DELETE_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/03/07 19:05:47
```

The display shows that the label of the sysplex member (in our case: SC31) is not present anymore (see Example 10-29).

Example 10-29 Display of the zones on DNS

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 858
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
```

```
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/03/07 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : NOT_PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/03/07 18:58:26
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : NOT_PRESENT            1
   TTL           : 60
   CLASS         : IN
   TYPE          : A
  RDATA         : 10.1.1.20
  GWM LABEL     : ZOS_LBA_ADVISOR
  GROUP LABEL   : ITSO_SYSPLEX
  LAST UPDATE   : 10/03/07 18:58:26
```

1 The current status, NOT_PRESENT, indicates that there is no connectivity to SC31.

Verify that you cannot ping the sysplex member DNS name

Try a **ping** command again (from SC32 in our environment):

```
ping WTSC31.example.com
```

This produced the results shown in Example 10-30.

Example 10-30 Ping with no success after we stopped the TCPIPC on SC31.

```
ping WTSC31.example.com
EZZ3111I Unknown host 'WTSC31.EXAMPLE.COM'
```

Re-start the TCP/IP stack and display the zones again

Start the TCP/IP stack on the sysplex member (in our case: SC31). This action will cause the IP address to become available. After starting the TCP/IP stack, issue the following command:

```
F ADNR,DISP,DNS,ZONES,DETAIL
```

You may initially see output similar that shown in Example 10-31. The process takes several seconds to update the DNS server and you might see *Update-Add_in_Progress* as intermediate status. If you issue the command again after a short time you should see the results similar to that shown in Example 10-32.

Example 10-31 Starting the TCP/IP stack and issuing display of the DNS RR status

```
S TCPIPB
$HASP100 TCPIPB ON STCINRDR
IEF695I START TCPIPB WITH JOBNAME TCPIPB IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
```

```
$HASP373 TCPIPB STARTED
EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPB ARE AVAILABLE.

F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 909
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/03/07 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : UPDATE-ADD_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/03/07 18:58:26
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : UPDATE-ADD_IN_PROGRESS
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : 10/03/07 18:58:26
```

Example 10-32 DNS RR status changes to PRESENT

```
F ADNR,DISP,DNS,ZONES,DETAIL
EZD1254I DNS ZONE DETAIL 847
DNS LABEL        : TEST85DNS
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
 ZONES ACTIVE    : 1
 ZONE LABEL      : MYZONE
  ZONE STATUS    : SYNCHRONIZED
  DOMAIN SUFFIX  : TEST.COM.
  ZONE TIMESTAMP : 10/03/07 18:51:39
  TSIG FLAGS     :
  DNS RR LABEL   : SC3XGROUP
   DNS RR STATUS : PRESENT
   TTL           : 60
   CLASS         : IN
   TYPE          : A
   RDATA         : 10.1.1.20
   GWM LABEL     : ZOS_LBA_ADVISOR
   GROUP LABEL   : ITSO_SYSPLEX
   LAST UPDATE   : N/A
  DNS RR LABEL   : WTSC31
   DNS RR STATUS : PRESENT
```

Archived

```
TTL           : 60
CLASS         : IN
TYPE          : A
RDATA         : 10.1.1.20
GWM LABEL     : ZOS_LBA_ADVISOR
GROUP LABEL   : ITSO_SYSPLEX
LAST UPDATE   : N/A
```

Verify you can ping the sysplex member DNS name again

We issued the following command from SC32:

```
ping WTSC31.example.com
```

and received the output shown in Example 10-33.

Example 10-33 Ping from the SC32 system against the name registered on DNS server through ADNR

```
Pinging host WTSC31.EXAMPLE.COM (10.1.1.20)
Ping #1 response took 0.000 seconds.
```

10.5 Problem determination for DNS service

This section covers the problem determination procedures for the caching-only DNS server and those for ADNR.

10.5.1 Problem determination for a caching-only DNS server

DNS server problems can be managed by using the following procedures:

- Review startup error messages.
- Review syslogd to obtain early logging messages.
- Start named from the OMVS shell to obtain early logging messages.
- Set debug and trace levels.
- Use the OMVS rndc command to manage the DNS daemon.

Review startup error messages

BPXBATCH does not handle a split parameter string that is created by putting single quotes around multiple JCL records and continuing them with a comma. If the PARM string is long enough to be continued onto the next line, you must code the parameter string contiguously to column 71, placing a continuation character in column 72, and then continuing the parameter string in column 16 of the next line. The unsupported format and resulting error message is shown in Example 10-34.

Example 10-34 NAMED9 JCL: example of invalid PARM string and resulting message at startup

```
BROWSE    SYS1.PROCLIB(NAMED9B) - 01.00              Line 00000000 Col 001 080
//NAMED9B  PROC PARMS='-d 0'
//NAMED9B  EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
// PARM=('PGM /usr/lpp/tcpip/sbin/named -V v9 ',
//       ' -c /etc/dnsdatb&SYSCLONE./named9b.conf &PARMS')
//STDENV   DD PATH='/etc/dnsdatb&SYSCLONE./named9b.env',
//            PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
```

```
//STDERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*

J E S 2  J O B  L O G  --  S Y S T E M  S C 3 1 --  N O D E
20.34.19 STC08336 ---- MONDAY,    24 Sep 2007 ----
20.34.19 STC08336  IEF695I START NAMED9X  WITH JOBNAME NAMED9X  IS ASSIGNED TO U
20.34.19 STC08336  $HASP373 NAMED9X  STARTED
20.34.20 STC08336  EZZ9089I EXITING NAMED, BIND V9 (DUE TO EARLY FATAL ERROR)
20.34.20 STC08336  $HASP395 NAMED9X  ENDED

The correct PARM field should look like this:
// PARM=('PGM /usr/lpp/tcpip/sbin/named -V v9 -c /etc/dnsdatb&SYSCLONE.X
//             /named9b.conf &PARMS')
```

Review syslogd to obtain early logging messages

For the BIND 9 name server, initial startup messages go to syslog. Later messages will be directed to other defined or default logs according to logging statements found or implied in the configuration file. For descriptions of the syslog file and the syslog daemon, see Chapter 1, “Syslogd” on page 1.

Start named from the OMVS shell to obtain early logging messages

The USERID that starts NAMED from the shell must have UID(0). Make sure your user ID either has UID(0) or that you can switch to superuser by entering the **su** command before you start NAMED.

If in your environment, your NAMED task requires environment variables, you can use the same environment variables file in the shell that is used by the started task JCL. Simply use the **export** command to set the file name containing the environment variables.

When you start **named** from the shell, you can specify a parameter of -g to ask for detailed logging. This can be beneficial if BPXBATCH or NAMED have problems initializing. You may miscode a parameter or use invalid syntax on one or more of the configuration statements, which causes either of the programs to terminate before they have progressed far enough to understand your logging options and directives. Use of the -g parameter overrides any logging (default or explicit) directives, and forces messages to be displayed to your shell session.

The use of the **su**, **export**, and the **named** commands is shown in Example 10-35.

Example 10-35 Starting NAMED9 from the z/OS UNIX shell to get early logging information

```
su

export _CEE_ENVFILE="/etc/dnsdatb31/named9b.env"

named -V v9 -g -c /etc/dnsdatb31/named9b.conf
```

Set debug and trace levels

Debugging levels can be specified at NAMED startup by using the -d parameter. They can later be changed in the configuration file, causing the server to reread the file. The BIND 9 name server relies on a start option, rndc, or the configuration file to define and alter the debug level. You can change the logging options in named.conf to gather more information, and then issue the *SIGHUP* signal or ’**rndc reload**’ to have the new logging options take effect. However, the preferred method is by using ’**rndc trace level**’.

Use the OMVS rndc command to manage the DNS daemon

The **rndc** utility can be used to provide a variety of functions that can be helpful in debugging name server problems. For example, the name server's cache can be viewed using the *dumpdb* parameter, and debug trace can be turned on or off using the *trace* parameter. If you suspect your cache is corrupted, you can flush the name server's cache with the *flush* parameter.

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

10.5.2 Problem determination for ADNR

If there are problems with ADNR, try the following problem determination techniques:

- Check that DNS, LBA and ADNR are running
- Check that the GWM is active
- Check that the sysplex members are available
- Check that the DNS status is active
- More problem determination tests for ADNR

Check that DNS, LBA and ADNR are running

Verify that the started task of Load Balance Advisor and ADNR are running. In addition, verify that the DNS server is running.

Check that the GWM is active

Issue the following console command:

```
F ADNR,DISP,GWM,DETAIL
```

Example 10-36 Display the GWM status

```
F ADNR,DISP,GWM,DETAIL
...
 EZD1254I GWM DETAIL 953
 GWM LABEL        : ZOS_LBA_ADVISOR
  GWM STATUS      : GWM_ACTIVE
  GWM TIMESTAMP   : 10/03/07 18:51:39
  GWM IPADDR..PORT: 10.1.9.29..3860
  LOCAL IPADDR    : 10.1.1.30
  UUID            : IBM_SYSPLEX_ADNR
  UPDATE INTERVAL : 60
  LAST UPDATE     : 10/03/07 19:07:47
 1 OF 1 RECORDS DISPLAYED
```

You may initially see the GWM STATUS value as CONVERGENCE_PENDING, but after a short time the status will change to GWM_ACTIVE, if ADNR is functioning. When the status changes to GWM_ACTIVE, it means that any updates from the GWM are reflected in the name server.

Check that the sysplex members are available

Issue the following console command:

```
F ADNR,DISP,GROUPS,GROUPID=itso_sysplex,DETAIL
```

This should produce results similar to those shown in Example 10-37.

Example 10-37 Display of the GROUP members availability status

```
F ADNR,DISP,GWM,GROUPS,GROUPID=ITSO_SYSPLEX,DETAIL
EZD1254I GWM GROUP DETAIL 950
GWM LABEL        : ZOS_LBA_ADVISOR
 GWM STATUS      : GWM_ACTIVE
 GWM TIMESTAMP   : 10/03/07 18:51:39
 GWM IPADDR..PORT: 10.1.9.29..3860
 LOCAL IPADDR    : 10.1.1.30
 UUID            : IBM_SYSPLEX_ADNR
 UPDATE INTERVAL : 60
 LAST UPDATE     : 10/03/07 19:07:47
 GROUP LABEL     : ITSO_SYSPLEX
  GROUP NAME     : SC3XGROUP
  GROUP TYPE     : HOST
  DNS LABEL      : TEST85DNS
  ZONE LABEL     : MYZONE
  MEMBER HOSTNAME:
   IPADDR        : 10.1.1.20
    AVAIL        : NO
    FLAGS        : NOTARGETSYS
    UPDATE COUNT : 4
  MEMBER HOSTNAME: WTSC31
   IPADDR        : 10.1.1.20
    AVAIL        : NO
    FLAGS        : NOTARGETSYS
    UPDATE COUNT : 4
```

The name is registered with the DNS server is WTSC31.example.com, where the first part of the name (WTSC31) came from the member hostname. The remainder of the name is from domain_suffix defined on the zone statement as shown in Example 10-19 on page 296.

itso_sysplex is the host_group name from the ADNR configuration file. Verify that under MEMBER HOSTNAME, the AVAIL field has the value YES. If the value is NO, check the configuration file of ADNR and update the IPADDR statement of the IP address which is not available to have an IP address that is configured under in the LBA configuration file.

Note: To enable ADNR to connect to the Advisor, the source IP address that ADNR uses to connect to the Advisor must be configured in the Advisor's lb_id_list statement. For information about the lb_id_list statement, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Check that the DNS status is active

Issue the following console command:

```
F ADNR,DISP,DNS,DETAIL
```

The results are similar to Example 10-38. This provides information about the name server that ADNR is updating.The DNS status should be ACTIVE and you should have at least one active zone.

Example 10-38 Displaying the DNS status

```
F ADNR,DISP,DNS,DETAIL
EZD1254I DNS DETAIL 915
DNS LABEL        : DNSSERV
 DNS STATUS      : ACTIVE
 DNS IPADDR..PORT: 10.1.2.99..53
 ZONES DEFINED   : 1
```

```
 ZONES ACTIVE    : 1
1 OF 1 RECORDS DISPLAYED
```

More problem determination tests for ADNR

If the ADNR process does not work, try the following steps to find the problem:

1. Enable high level of logging for ADNR by issuing the following command:

   F ADNR,DEBUG,LEVEL=32

2. Verify that you get messages sent to syslogd:

   a. Verify that syslogd is up by issuing **ps –ef | grep syslogd** under USS.

   b. Verify that the configuration file for syslogd is configured correctly.

   c. If you make any change to the syslogd configuration file, stop and start syslogd. You can do it by issuing a **kill** command stop syslogd and issuing the syslogd command to start syslogd. By default it uses the /etc/syslog.conf configuration file.

   d. Verify that you have enough free space in the directory where the log file resides.

   > **Note:** The ADNR log file is very helpful for solving problems.

3. Verify that the DNS server version is at least 9.2. You can do it by issuing the following command from the DNS server:

   named –v

4. Enable logging on your DNS server. You may do it by updating the DNS configuration file (usually it will be /etc/named.conf) to include the following line, as shown in Example 10-39.

Example 10-39 Enabling high logging level on the DNS server

```
logging {
     channel dnslog {
          file "/var/named/bind9.log" versions 3 size 3m;
          severity debug 10;
          print-time yes;
          print-severity yes;
          print-category yes;
          };
     category default {
          dnslog;
          };
     };
```

   In this example the log of the DNS is written to /var/named/bind9.log. After you update the configuration file, restart the DNS server.

5. From the DNS server, issue the following command:

   tail -f /var/named/bind.log

   This command displays the last lines of the log file and outputs appended data as the file grows. You may use this command to see if the z/OS image is trying to send requests to the DNS server. You should leave the tail command running till you are satisfied that ADNR is working.

6. Verify that you can issue **nslookup** from the z/OS image to the specific DNS server by issuing:

```
nslookup
```

And from the nslookup program issue the following command:

```
server 10.1.2.99
```

Where 10.1.2.99 represents the DNS server IP address. You are supposed to get the following screen:

```
   Default Server:  dnsserv.example.com
   Address:  10.1.2.99
>
```

Type **exit** to end the nslookup program.

Verify that the user who runs the ADNR started task has full permissions on its home directory. If the permissions are insufficient you might see the message shown in Example 10-40.

Example 10-40 Console error message

```
BPXF903I THE ATTRIBUTE RETRIEVAL CALL (IGWASMS) FOR FILE SYSTEM 576
.DIGRC.HFS FAILED.
RC = 00000008, RS = 00000008, DIAG = 0000000000000000
```

In addition, you will see the message shown in Example 10-41 in the ADNR log file.

Example 10-41 Error message in the ADNR log file

```
Zone transfer for zone myzone failed, rc = No file could be created
```

10.6 Additional information sources for DNS

Detailed information about the DNS server and protocol can be found in the following documents:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534

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

Archived

Archived

A

Environment variables

This appendix discusses environment variables used by the z/OS TCP/IP stack and associated applications.

Environment variables are named variables, with assigned values, that can be accessed by various processes in the z/OS Communications Server configuration. Applications use environment variables to define the characteristics of their specific environment. *z/OS Communications Server: IP Configuration Guide*, SC31-8775, provides details about where to find information about the environment variables that are explicitly set by the z/OS Communications Server and its applications. In addition to an application being able to set its own environment variables, Language Environment® and z/OS UNIX System Services (z/OS UNIX) also provide environment variables. The following discussion assists with determining which applications have access to these additional Language Environment and z/OS UNIX System Services variables.

Understanding the resolver search orders used in native MVS API environments and in z/OS UNIX environments is key to setting up your system properly. The type of API environment not only affects what search order is used by the resolver to locate certain files required for processing, but it also determines which set of environment variables is available to the resolver and to the server programs. You can indirectly determine which sets of environment variables an application can use by identifying the *caller API value* obtained from the output of a resolver trace performed when the application calls the resolver. For information about dynamically starting the resolver trace, refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

© Copyright IBM Corp. 2008. All rights reserved.

Description of the environment variable information

We summarized many details by organizing the information into tables. Table A-1 and Table A-2 on page 313 show sample commands and applications that use the different API interfaces. They list the most commonly used environment variables for the z/OS TCP/IP stack and associated applications. Some applications use only the z/OS UNIX System Services API, some use only the Language Environment API, some use both, and some use neither. In addition, some applications set their own environment variables.

The variables associated with the z/OS UNIX System Services API are listed in Table A-3 on page 313. The variables associated with the Language Environment API are listed in Table A-4 on page 314.

The variables specific to each application are listed in Table A-5 on page 315. The application-specific table also indicates which of the API interfaces are used by the application. When an application uses the indicated API interface, it also has access to that API's set of environmental variables.

The following sections discuss the API environments.

- "Native MVS API environment" on page 312
- "z/OS UNIX API environment" on page 313
- "z/OS UNIX System Services environment variables" on page 313
- "Language Environment, environment variables" on page 314
- "Application-specific environment variables" on page 315
- "Setting environment variables" on page 321

Native MVS API environment

The following *caller API values* indicate that the native *MVS API environment* search order is used, and access to the Language Environment and z/OS UNIX System Services environment variables is not available:

- TCP/IP C Sockets
- TCP/IP Pascal Sockets
- TCP/IP REXX Sockets
- TCP/IP Sockets Extended

Table A-1 lists examples of commands and applications that use the native *MVS API environment*.

Table A-1 Examples of commands and applications that use the native MVS API environment

TSO commands	Applications
DIG LPR NETSTAT NSLOOKUP PING REXEC RPCINFO RSH TRACERTE	CICS Listener LPD Miscellaneous server PORTMAP RSHD (RXSERVE) SMTP TN3270

z/OS UNIX API environment

The following *caller API values* indicate that the z/OS UNIX API environment search order is used, and the Language Environment and z/OS UNIX System Services environment variables are available:

- Language Environment C Sockets
- UNIX System Services

Table A-2 lists some examples of commands and applications that use the z/OS UNIX environment.

Table A-2 Examples of commands and applications that use the z/OS UNIX environment

z/OS UNIX command	Applications
dig dnsdomainname domainname ftp host hostname netstat nslookup ping rexec rsh rpcinfo sendmail snmp traceroute	FTP SNMP agent z/OS UNIX OPORTMAP z/OS UNIX OREXECD z/OS UNIX ORSHD z/OS UNIX RPCBIND

z/OS UNIX System Services environment variables

z/OS UNIX System Services Planning, GA22-7800, contains an appendix that discusses environment variables (_BPX_ and _BPXK_) used by the z/OS UNIX System Services kernel. Applications that use the z/OS UNIX System Services API have access to these variables. These are not specific to TCP/IP functions and are listed here for completeness.

Table A-3 z/OS UNIX System Services API environment variables

z/OS UNIX System Services API environment variable	Description
_BPX_ACCT_DATA	Sets accounting information for the caller
_BPX_BATCH_SPAWN	Specifies whether BPXBATCH is to use spawn instead of fork
_BPX_BATCH_UMASK	Sets the permission bits for a file
_BPX_JOBNAME	Sets the job name of a process by overriding the default
_BPX_PTRACE_ATTACH	Used for debugging target programs
_BPX_SHAREAS	Allows a spawned child process to share the shell's address space
_BPX_SPAWN_SCRIPT	Flags the specified file as a shell script
_BPX_TERMPATH	Determines the origin of a logged user
_BPX_UNLIMITED_SPOOL	Can limit spooled output

Archived

_BPX_USERID	Sets the user ID of a spawned process
_BPXK_AUTOCVT	Controls automatic file conversion
_BPXK_CCIDS	Defines a pair of coded character set IDs
_BPXK_DAEMON_ATTACH	Controls the security environment of RACF-DELEGATED resources
_BPXK_INET_FASTPATH	Used by TCP/IP to access FASTPATH mode
_BPXK_JOBLOG	Controls writing WTO messages to a job log file
_BPXK_MDUMP	Controls destination of SYSMDUMP output
_BPXK_SETIBMOPT_TRANSPORT	Sets stack affinity
_BPXK_WLM_PROPAGATE	Controls propagation of WLM enclaves

Language Environment, environment variables

z/OS XL C/C++ Programming Guide, SC09-4765, contains information about some of the environment variables reserved for z/OS XL C/C++. Applications that use the Language Environment API have access to these variables.

Table A-4 Language Environment API environment variables

LE Environment variable	Description
_CEE_DLLLOAD_XPCOMPAT	Controls z/OS DLL load order
_CEE_DMPTARG	Specifies directory for Language Environment dumps (CEEDUMPs)
_CEE_ENVFILE	Points to a file containing other environment variables
_CEE_HEAP_MANAGER	Specifies the Vendor Heap Manager (VHM) DLL name
_CEE_RUNOPTS	Sets Language Environment runtime options
_EDC_ADD_ERRNO2	Appends errno2 information to the output of perror() and strerror()
_EDC_ANSI_OPEN_DEFAULT	Affects characteristics of z/OS text files opened with default attributes
_EDC_BYTE_SEEK	Indicates that ftell() should return relative byte offsets
_EDC_CLEAR_SCREEN	Affects the clearing of output screens
_EDC_COMPAT	Controls compatibility with old C/370™ code
_EDC_C99_NAN	Controls binary floating-point representation
_EDC_ERRNO_DIAG	Controls additional diagnostic information
_EDC_GLOBAL_STREAMS	Controls the stdin, stdout, and stderr data streams
_EDC_POPEN	Uses fork() or spawn() to create a child process
_EDC_PUTENV_COPY	Sets the behavior of the putenv() function
_EDC_RRDS_HIDE_KEY	Applies to VSAM RRDS files opened in record mode
_EDC_STOR_INCREMENT	Controls storage increments above the 16M line
_EDC_STOR_INCREMENT_B	Controls storage increments below the 16M line
_EDC_STOR_INITIAL	Sets initial size of library storage above the 16M line

_EDC_STOR_INITIAL_B	Sets initial size of library storage below the 16M line
_EDC_ZERO_RECLEN	Allows processing of zero-length records in a z/OS variable file

Application-specific environment variables

The application-specific environment variables are listed in Table A-5.

The first table entry for each application indicates whether the application has access to the z/OS UNIX API variables referenced in Table A-3 on page 313. When the z/OS UNIX API variables entry indicates `Yes`, the application has access to the z/OS UNIX API environment variables.

The second table entry for each application indicates whether the application has access to the Language Environment API variables referenced in Table A-4 on page 314. When the Language Environment API variables entry indicates `Yes`, the application has access to the Language Environment API environment variables.

The remaining table entries for each application show the application's specific environment variables.

Table A-5 Application-specific environment variables

Application-specific environment variable	Description
RESOLVER	
z/OS UNIX API variables	Has indirect access to these via program call
Language Environment API variables	Has no access to these
RESOLVER_IPNODES	Points to the /etc/ipnodes file
X_ADDR	Points to hlq.HOSTS.ADDRINFO
X_SITE	Points to hlq.HOSTS.SITEINFO
HOSTALIASES	Points to the host alias name file
RESOLVER_TRACE	Points to the file into which the resolver trace output is written
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
X_XLATE	Points to hlq.STANDARD.TCPXLBIN
General stack environment configuration	
z/OS UNIX API variables	Has indirect access via program call
Language Environment API variables	Has indirect access via program call
LOCALDOMAIN	Overrides any other setting for domain within TCPDATA
MESSAGECASE	Overrides any other setting for message case within TCPDATA
SYSLOGD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313

Archived

Language Environment API variables	Has access to the variables in Table A-4 on page 314
IKED	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
IKED_FILE	Configuration file for IKED
IKED_CTRACE_MEMBER	Parmlib member containing CTRACE settings for IKED
Load Balancing Advisor and Agent	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
TN3270E Telnet server	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 313
Language Environment API variables	No access to the variables in Table A-4 on page 314
INETD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
OTELNETD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
FTP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
KRB5_SERVER_KETAB	Kerberos security setting
_FTPXLATE_name	(CCXLATE name) Translate table for control connection
_FTPXLATE_name	(XLATE name) Translate table for data connection
_ICONV_UCS2	Conversion methods for FTP
TFTPD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313

Language Environment API variables	Has access to the variables in Table A-4 on page 314
TSO REXEC	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 313
Language Environment API variables	No access to the variables in Table A-4 on page 314
UNIX REXECD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
UNIX RSHD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
_EUV_SVC_STDOUT_FILENAME	Sets default message output file name
_EUV_SVC_DBG_FILENAME	Overrides default file: Debug messages written to this file
_EUV_SVC_DBG_MSG_LOGGIN	Specifies whether to generate debug message
_EUV_SVC_DBG_TRACE	Specifies whether to generate trace messages
_EUV_SVC_MSG_FACILITY	Facility class setting
_EUV_SVC_MSG_LOGGING	Logging method
KRB5_SERVER_KETAB	Kerberos security table
LANG	Local language and customs
LC_COLLATE	Character collation
LC_CTYPE	Character handling
LC_MESSAGES	Message handling
LC_MONETARY	Currency formatting
LC_NUMERIC	Numeric formatting
LC_TIME	Date and time formatting
LC_ALL	Sets all the above LC_ variables with one setting
NLSPATH	Locates message catalogs
PATH	Default path(s) for locating files
SMTP	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 313
Language Environment API variables	No access to the variables in Table A-4 on page 314

Archived

Sendmail	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
GSK_TRACE	Specifies a bit mask enabling system SSL trace options.
GSK_TRACE_FILE	When set to the name of a file, enables the system SSL trace.
HOME	The path name of the user's home directory.
HOSTALIASES	The host aliases file for sendmail.
LPD/LPR	
z/OS UNIX API variables	No access to the variables in Table A-3 on page 313
Language Environment API variables	No access to the variables in Table A-4 on page 314
DHCP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
LPR_PRINTER	Output printer for DHCP reports
BIND4	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
DNS_VERSION	Version of Bind (V4 or V9) Default setting
HOSTALIASES	Host aliases file
PAGER	VIEW subcommand filter
BIND9	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
DNS_VERSION	Version of Bind (V4 or V9) Default setting
OMPROUTE	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
OMPROUTE_FILE	Configuration file for Omproute
OMPROUTE_DEBUG_FILE	Debug output file for Omproute

OMPROUTE_DEBUG_FILE_CONTROL=1000,5	Omproute's Debug file size
OMPROUTE_OPTIONS=hello_hi	Options for Omproute
OMPROUTE_CTRACE_MEMBER=CTIORAT0	CTRACE options for Omproute
SNMP_PORT	SNMP subagent listens on this port
TMPDIR	Directory for temporary work files
SNMP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
OSNMP_CONF	Configuration file for `osnmp`/`snmp` command /etc/snmpv2.conf
SNMPD_CONF	Configuration file for SNMP agent (V3) /etc/snmpd.conf
MIBS_DATA	MIB descriptions for SNMP /etc/mibs.data
OSNMPD_DATA	Control file for SNMP agent /etc/osnmpd.data
PW_SRC	Community security file for SNMP (V2) /etc/pw.src
SNMPD_BOOTS	Boot file for SNMPD agent (v3) /etc/snmpd.boots
SNMPTRAP_DEST	Control file for SNMPTRAP (v2) /etc/snmptrap.dest
TRAPFWD_CONF	Configuration file for TRAPFWD /etc/trapfwd.conf
SNMPQE	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
MIB_DESC	MIB description file for SNMPQE /etc/mibdesc.data
SLAP	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
SNMP_PORT	SNMP agent listens on this port
Policy Agent	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
TZ	Local time zone
PAGENT_CONFIG_FILE	Configuration file for policy agent /etc/pagent.conf

Archived

PAGENT_LOG_FILE	Logfile for policy agent
PAGENT_LOG_FILE_CONTROL	Policy Agent's logfile size
RSVP Agent	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
RSVPD_CONFIG_FILE	Configuration file for RSVP /etc/rsvpd.conf
TRMD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
RESOLVER_CONFIG	Resolver configuration file /etc/resolv.conf (tcpdata)
TZ	Local time zone
LIBPATH	Program execution library /usr/lib
TIMED	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
SNTPD	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
PORTMAPPER	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
UNIX PORTMAPPER	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
MISCSERV	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314

DCAS	
z/OS UNIX API variables	Has access to the variables in Table A-3 on page 313
Language Environment API variables	Has access to the variables in Table A-4 on page 314
DCAS_CONFIG_FILE	Configuration file /etc/dcas.conf for DCAS

Setting environment variables

Setting an environment variable so that a z/OS UNIX application can retrieve the value depends on whether the z/OS UNIX application is started from the z/OS shell or from JCL. If the z/OS UNIX application is to be started from the z/OS shell, the export shell command can be used to set the environment variable. For example, to set the value of RESOLVER_CONFIG to /etc/tcpa.data, you can code the following export command:

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

If, instead of a UNIX file you want to set RESOLVER_CONFIG to the data set MVSA.PROD.PARMS(TCPDATA), then you can specify the following export command. Be certain to put the single (' ') quotation marks around the data set name. If you do not, your user ID will be added as a prefix to the data set name when the resolver tries to open the file.

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

If the z/OS UNIX application is to be started from JCL instead of from the z/OS shell, the environment variable needs to be passed as a parameter in the JCL for the application. For example, the following example shows the RESOLVER_CONFIG variable pointing to a UNIX file:

```
//OSNMPD PROC
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/',
//      'ENVAR("RESOLVER_CONFIG=/etc/tcpa.data")/-d 0')
```

The following example shows the RESOLVER_CONFIG variable pointing to a PDS member:

```
//OSNMPD    PROC
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/',
//  'ENVAR("RESOLVER_CONFIG=//''TCPA.MYFILE(TCPDATA)'")/-d 0')
```

The following example shows the RESOLVER_CONFIG variable pointing to the TCPIP.DATA information from a DD card:

```
//OSNMPD PROC
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/',
// 'ENVAR("RESOLVER_CONFIG=DD:TCPDATA")/-d 0')
//TCPDATA DD DSN=TCPA.MYFILE(TCPDATA),DISP=SHR
```

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

```
//OSNMPD    PROC
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/',
// 'ENVAR("_CEE_ENVFILE=DD:STDENV")/-d 0')
//STDENV DD DSN=TCPA.MYFILE(ENVFILE),DISP=SHR
```

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

> **Important:** If this file is a z/OS data set (traditionally referred to as an MVS data set), the data set must be allocated with RECFM=V.
>
> Using RECFM=F is *not* recommended, because RECFM=F enables padding with blanks for the environment variables. In this case, if an environment variable is specifying a UNIX file name, the resulting name includes the trailing blanks of the record, and the system cannot find the named file. Error messages are issued by the program attempting to locate the file.

See *z/OS XL C/C++ Programming Guide*, SC09-4765, for more information about specifying a list of environment variables using the _CEE_ENVFILE environment variable pointer.

Archived

B

Sample files provided with TCP/IP

This appendix lists the common sample files that are provided with the z/OS Communications Server product. These samples can be found in either of two locations:

- SYS1.SEZAINST, an MVS PDS
- /usr/lpp/tcpip/samples/, an HFS directory

© Copyright IBM Corp. 2008. All rights reserved.

Sample files by component

Table B-1 Sample files provided with TCP/IP components

Resolver	
SEZAINST(RESOPROC)	JCL for the resolver proc
SEZAINST(RESSETUP)	Resolver setup file
General stack environment configuration	
SEZAINST(TCPIPROC)	JCL for the TCP/IP proc
SEZAINST(SAMPPROF)	PROFILE.TCPIP file for TCP/IP
SEZAINST(TCPDATA)	TCPIP.DATA file for TCP/IP
SEZAINST(IPNODES)	*hlq*.ETC.IPNODES file
SEZAINST(HOSTS)	HOSTS.LOCAL (or /etc/hosts) file
SEZAINST(CONVSYM)	symbol translator JCL job
SEZAINST(EZARACF)	SAF authorizations
SEZAINST(PROTO)	hlq.ETC.PROTO file
/usr/lpp/tcpip/samples/protocol	/etc/protocol file
SEZAINST(SERVICES)	ETC.SERVICES
/usr/lpp/tcpip/samples/services	/etc/services file
OMPROUTE	
SEZAINST(OMPROUTE)	JCL for OMPROUTE proc
SEZAINST(EZAORCFG)	OMPROUTE configuration file
SYSLOGD	
SEZAINST(SYSLOGD)	JCL for SYSLOGD proc
/usr/lpp/tcpip/samples/syslog.conf	SYSLOGD configuration file
/usr/lpp/tcpip/samples/rmoldlogs	Remove Old Logs script
TN3270E Telnet server	
SEZAINST(EZBTNPRC)	JCL for TN3270E Telnet server proc
SEZAINST(TNPROF)	TN3270 configuration file
SEZAINST(EZBTPUST)	3270 data stream USS table
SEZAINST(EZBTPSCS)	SCS data stream USS table
SEZAINST(EZBTPINT)	INTERPRET table
SEZAINST(EZBUSJCL)	JCL to assemble and link a USS table

Archived

SEZAINST(TNDBCSCN)	Double byte transform support
SEZAINST(VTAMLST)	VTAMLST definitions
SEZAINST(IVPLU)	VTAM verification procedure for Telnet
/usr/lpp/tcpip/samples/mvstn3270.mi2	mibs for TN3270
OTELNETD	
/usr/lpp/tcpip/samples/services	/etc/services file
/usr.lpp/tcpip/samples/inetd.conf	/etc/inetd.conf file
INETD	
/usr/lpp/tcpip/samples/services	/etc/services file
/usr.lpp/tcpip/samples/inetd.conf	/etc/inetd.conf file
FTP	
SEZAINST(FTPD)	JCL for FTPD proc
SEZAINST(FTCDATA)	FTP.DATA for the Client
SEZAINST(FTPSDATA)	FTP.DATA for the Server
TFTPD	
SEZAINST(TFTPD)	JCL for the TFTPD proc
IKED	
SEZAINST(IKED)	JCL for the IKED proc
/usr/lpp/tcpip/samples/iked.conf	Configuration file for IKED
SMTP	
SEZAINST(SMTPPROC)	JCL for SMTP proc
SEZAINST(SMTPCONF)	SMTP configuration file
SEZAINST(SMTPNOTE)	REXX source for SMTPNOTE
SEZAINST(SMTPEXIT)	SMTP exit controls “spam” mail
Sendmail	
SEZAINST(SENDMAIL)	JCL for sendmail proc
/usr/lpp/tcpip/samples/sendmail/cf/sample.cf	Configuration file for MTA and MUA
/usr/lpp/tcpip/samples/sendmail/cf/submit.cf	Configuration file for MUA specific

Archived

/usr/lpp/tcpip/samples/sendmail/cf/submit.mc	Input master configuration file
/usr/lpp/tcpip/samples/sendmail/sendmail.ps	Install and Ops Guide for sendmail 8.12
/usr/lpp/tcpip/samples/sendmail/cf/zOS.cf	z/OS-specific SSL for MTA
/usr/lpp/tcpip/samples/sendmail/ostype/zOS.m4	Describes the z/OS environment
/usr/lpp/tcpip/samples/sendmail/README.m4	Latest Instructions for sendmail
/usr/lpp/tcpip/samples/sendmail/feature/msp.m4	Special features msp file for sendmail
LPD/LPR	
SEZAINST(LPSPROC)	JCL for LPD server proc
SEZAINST(LPDDATA)	LPD.CONFIG configuration file
SEZAINST(EZAAE04S)	LPBANNER source file
SEZAINST(EZAAE04T)	LPBANNER translate table
BIND4	
SEZAINST(NAMED4)	JCL for BIND 4 Proc
/usr/lpp/tcpip/samples/named.boot	Boot file for bind4
/usr/lpp/tcpip/samples/db.mycorp.v4	Sample version 4 file
/usr/lpp/tcpip/samples/db.34.37.9.v4	Sample version 4 file
/usr/lpp/tcpip/samples/db.loopback.v4	Sample version 4 file
/usr/lpp/tcpip/samples/slave.boot	Boot file for a slave server
/usr/lpp/tcpip/samples/caching.boot	Boot file for a caching server
BIND9	
SEZAINST(NAMED9)	JCL for BIND 9 Proc
/usr/lpp/tcpip/samples/named.conf	Boot file for bind9
/usr/lpp/tcpip/samples/db.mycorp.v9	Sample version 9 file
/usr/lpp/tcpip/samples/db.34.37.9.v9	Sample version 9 file
/usr/lpp/tcpip/samples/db.loopback.v9	Sample version 9 file
/usr/lpp/tcpip/samples/slave.conf	Boot file for a slave server
/usr/lpp/tcpip/samples/caching.conf	Boot file for a caching server
SNMP	
SEZAINST(OSNMPDPR)	JCL for osnmpd proc
SEZAINST(SNMPPROC)	JCL for snmpqe proc
SEZAINST(MIBDESC)	*hlq*.MIBDESC.DATA file

Archived

SEZAINST(EZASNTPR)	JCL for trap forwarder proc TRAPFWD
SEZAINST(TRAPFWD)	JCL for trap forwarder proc
SEZAINST(MSSNMP)	NLS Message data set for SNMPQE
/usr/lpp/tcpip/samples/mvstcpip.caps	SNMP agent capabilities statement
/usr/lpp/tcpip/samples/mvstcpip.mi2	IBM enterprise-specific mib
/usr/lpp/tcpip/samples/mvstn3270.mi2	TN3270 mib file
/usr/lpp/tcpip/samples/mibs.data	Textual names user mib file
/usr/lpp/tcpip/samples/snmpv2.conf	Configuration file for SNMPV2
/usr/lpp/tcpip/samples/snmpd.conf	Configuration file for SNMP
/usr/lpp/tcpip/samples/osnmpd.data	Control data file for SNMP
SNMPQE	
SEZAINST(SNMPPROC)	JCL for SNMPQE proc
REXEC	
SEZAINST(RXPROC)	JCL for REXECD proc
SEZAINST(RXUEXIT)	User exit for REXEC
UNIX REXECD	
/usr/lpp/tcpip/samples/syslog.conf	SYSLOGD configuration file
/usr/lpp/tcpip/samples/services	/etc/services file
/usr.lpp/tcpip/samples/inetd.conf	/etc/inetd.conf file
/usr/sbin/orexecd	location of program orexecd
/usr/lib/nls/msg/C/rexdmsg.cat	message catalog for rexecd
UNIX RSHD	
/usr/lpp/tcpip/samples/syslog.conf	SYSLOGD configuration file
/usr/lpp/tcpip/samples/services	/etc/services file
/usr.lpp/tcpip/samples/inetd.conf	/etc/inetd.conf file
/usr/sbin/orshd	Location of program orshd
/usr/sbin/ruserok	User verification
/usr/lib/nls/msg/C/rshdmsg.cat	Message catalog file for rshd
TIMED	
SEZAINST(TIMED)	JCL for TIMED proc

Archived

SNTPD	
SEZAINST(SNTPD)	JCL for SNTPD proc
PORTMAPPER	
SEZAINST(PORTPROC)	JCL for PORTMAPPER proc
SEZAINST(ETCRPC)	ETC.RPC control file
UNIX PORTMAPPER	
SEZAINST(OPORTPRC)	JCL for UNIX portmapper proc
RPCBIND	
SEZAINST(RPCBIND)	JCL for RPCBIND proc
/etc/services	Port reservation file for services
NCS INTERFACE	
SEZAINST(NRGLBD)	JCL for NCS Global Location Broker
SEZAINST(LLBD)	JCL for Local Location Broker
MISCSERV	
SEZAINST(MISCSERV)	JCL for Miscellaneous server proc
SEZAINST(MSMISCSR)	NLS Message data set for MISC Server
DCAS	
SEZAINST(EZADCASP)	Alias JCL for DCAS proc
SEZAINST(DCAS)	JCL for DCAS proc
Load Balancing Advisor and Agent	
SEZAINST(EZBLBADV)	JCL for Advisor proc
SEZAINST(EZBLBADC)	JCL for Advisor configuration file
SEZAINST(EZBLBAGE)	JCL for Agent proc
SEZAINST(EZBLBAGC)	JCL for Agent configuration file
SEZAINST(LBADVCNF)	LBADV.CONF
SEZAINST(LBAGECNF)	LBAGENT.CONF

Archived

SLAP	
SEZAINST(EZAPAGSB)	Alias JCL for SLAPM2 subagent proc
SEZAINST(NSLAPM2)	JCL for SLAPM2 subagent proc
SEZAINST(EZAPAGSN)	Alias JCL for SLAP subagent proc
SEZAINST(PAGTSNMP)	JCL for SLAP subagent proc
usr/lpp/tcpip/samples/slapm2.mi2	MIB file for NSLAPM2
RSVP Agent	
SEZAINST(RSVPD)	JCL for RSVPD proc
/usr/lpp/tcpip/samples/rsvpd.conf	Configuration file for RSVPD
TRMD	
SEZAINST(TRMD)	JCL for TRMD proc
Policy Agent	
SEZAINST(EZAPAGSP)	Alias JCL for Policy Agent proc
SEZAINST(PAGENT)	JCL for Policy Agent proc
Policy Agent Configuration	
/usr/lpp/tcpip/samples/pagent.conf	Configuration file for Policy Agent
IPSec policy definitions	
/usr/lpp/tcpip/samples/pagent_CommonIPSec.conf	Configuration files: common IPSEc
/usr/lpp/tcpip/samples/pagent_IPSec.conf	Configuration files: stack-specific IPSec
AT-TLS policy definitions	
/usr/lpp/tcpip/samples/pagent_TTLS.conf	AT-TLS policy definitions
LDAP policy definitions	
/usr/lpp/tcpip/samples/pagent.ldif	LDAP top level directory structure
/usr/lpp/tcpip/samples/pagent_starter_QOS.ldif	Information file for QOS starter set
/usr/lpp/tcpip/samples/pagent_starter_IDS.ldif	Information file for IDS starter set
/usr/lpp/tcpip/samples/pagent_advanced_QOS.ldif	Information file for QOS advanced
/usr/lpp/tcpip/samples/pagent_advanced_IDS.ldif	Information file for IDS advanced

Archived

LDAP protocol version 2	
/usr/lpp/tcpip/samples/pagent_oc.conf	Object class definitions
/usr/lpp/tcpip/samples/pagent_at.conf	attribute Definitions
LDAP protocol version 3	
/usr/lpp/tcpip/samples/pagent_schema.ldif	V2 core and QOS object class
/usr/lpp/tcpip/samples/pagent_v3schema.ldif	V3 additions to V2 core
/usr/lpp/tcpip/samples/pagent_schema_updates.ldif	Updates to V2 core
/usr/lpp/tcpip/samples/pagent_idsschema.ldif	IDS schema file V3 for v1.4
/usr/lpp/tcpip/samples/pagent_qosschema.ldif	QOS schema file V3 for v1.5
/usr/lpp/tcpip/samples/pagent_r5idsschema.ldif	V3 IDS for v1.5
/usr/lpp/tcpip/samples/pagent_schema_r5updates.ldif	Updates for IDS v1.5
/usr/lpp/tcpip/samples/pagent_r6qosschema.ldif	QOS V3 for v1.6
/usr/lpp/tcpip/samples/pagent_schema_r6updates.ldif	Updates for V3 core for v1.6
Version 3 schema draft documents	
/usr/lpp/tcpip/samples/pagent_pcim.txt	Policy Core Information Model draft
/usr/lpp/tcpip/samples/pagent_core.txt	Policy Core LDAP draft for v1.2
/usr/lpp/tcpip/samples/pagent_cond.txt	Policy conditions draft for v1.2
C applications: policy performance monitoring	
/usr/lpp/tcpip/samples/pagent/README	Instructions for running the following pgms
/usr/lpp/tcpip/samples/pagent/pCollector.c	Test pgm: Policy API performance data
/usr/lpp/tcpip/samples/pagent/pCollector.h	Header file for pCollector pgm
/usr/lpp/tcpip/samples/pagent/pLogReader.c	Test pgm: reads policy performance log

Archived

C

Configuration files: TN3270E standalone started task scenario

This appendix contains the configuration files used for the TN3270E Telnet server standalone started task scenario.

© Copyright IBM Corp. 2008. All rights reserved.

SC30 files for TN3270 standalone started task scenario

The following sections are included:

- "SC30 TN3270B Server PROC for TN3270 standalone started task" on page 332
- "SC30 TN3270B Server PROFILE for TN3270 standalone started task" on page 332
- "SC30 TCPIPB stack PROC for TN3270 standalone started task" on page 334
- "SC30 TCPIPB stack PROFILE for TN3270 standalone started task" on page 335
- "SC30 OMPROUTE PROC for TN3270 standalone started task" on page 338
- "SC30 OMPROUTE STDENV file for TN3270 Standalone Task Scenario" on page 338
- "SC30 OMPROUTE CONFIG for TN3270 standalone started task" on page 338

SC30 TN3270B Server PROC for TN3270 standalone started task

Figure C-1 lists the JCL procedure for the SC30 TN3270B server.

```
 BROWSE    SYS1.PROCLIB(TN3270B) - 01.02             Line 00000000 Co
 Command ===>                                                  Scroll
********************************** Top of Data ************************
//TN3270B  PROC PARMS='TRC=TN',
//             PROFILE=TELNB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//TN3270B  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
******************************** Bottom of Data *********************
```

Figure C-1 SC30 TN3270B PROC JCL

We started the server task by issuing the following command:

```
S TN3270B,PROFILE=TELNB30A
```

SC30 TN3270B Server PROFILE for TN3270 standalone started task

The profile for the TN3270E Telnet server for this scenario is TELNB30A. The naming convention for the profiles is *XXXXyccs*:

- *XXXX* is TELN.
- *y* is B for the stack that was used for the examples in this book.
- *cc* is the &SYSCLONE value of this system.
- *s* is a letter representing the specific scenario (A, B, C, D).

```
 BROWSE    TCPIPB.TCPPARMS(TELNB30A) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
********************************** Top of Data ***********************************
; ===SC30============ TN3270E Telnet server Profile for Standalone Task =======-
;                                                                               -
; No SSL security.  No Sysplex Distribution in the stack.                       -
;        This is a plain server profile, very simple and easy.                  -
; ------------------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
; ------------------------------------------------------------------------------
; These default device type settings will be used by all ports if no   -
; TELNETPARMS or PARMSGROUP is used to override the settings.           -
; They are logmode names shipped in ISTINCDT with the latest level of  -
; VTAM.                                                                 -
; ------------------------------------------------------------------------------
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
    ;
ENDTELNETGLOBALS
;
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
```

Figure C-2 SC30 TN3270B profile (TELNB30A)

Archived

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL SC30N*          ; Netview
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
******************************** Bottom of Data ********************************
```

Figure C-3 SC30 TN3270B profile (TELNB30A) - end

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC30BB*nn*):

- *XXXX* is the &SYSNAME of the system.
- *y* is B for stack B that was used for all of the examples in this book.
- *z* is B for the basic port (23) and S for the secure port (992).
- *nn* is 01 through 99.

Because TN3270 maintains a master list of LU names in use across all ports, we could have used the same group of default LU names for all ports in our setup. We chose different pool names, one for basic and one for secure, to help us verify which port we are connected to in our display examples. Because there is only one port defined and in use in *this* scenario, all LU names will be shown as SC30BBnn.

SC30 TCPIPB stack PROC for TN3270 standalone started task

The JCL procedure for the stack is TCPIPB.

```
//TCPIPB    PROC PARMS='CTRACE(CTIEZB00),IDS=00',
//               PROFILE=PROFB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE
//TCPIPB    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//               PARM=('&PARMS',
//         'ENVAR("RESOLVER_CONFIG=//''TCPIPB.TCPPARMS(&TCPDATA)'''")')
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*TNDBCSER DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure C-4 SC30 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB30A
```

SC30 TCPIPB stack PROFILE for TN3270 standalone started task

The PROFILE for the stack is PROFB30A, as shown in Figure C-5, Figure C-6 on page 336, Figure C-7 on page 337, and Figure C-8 on page 338.

```
 BROWSE    TCPIPB.TCPPARMS(PROFB30A) - 01.06         Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
********************************* Top of Data ***********************************
;
; TCPIP.PROFILE.TCPIP     for Standalone TN3270 and no Sysplex Dist.
; ===================
;
GLOBALCONFIG
   ECSALIMIT 0M              ; default = 0M
   POOLLIMIT 0M              ; default = 0M
   TCPIPSTATISTICS           ; default = notcpipstatistics
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
   RESTRICTLOWPORTS          ; default = restrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   DELAYACKS                 ; default = delayacks
;
UDPCONFIG
   RESTRICTLOWPORTS          ; default = restrictlowports
   UDPCHKSUM                 ; default = udpchksum
   UDPSENDBFRSIZE   65535    ; default = 64K
   UDPRCVBUFRSIZE   65535    ; default = 64K
 NOUDPQUEUELIMIT             ; default = udpqueuelimit
;
 SMFCONFIG                    ;
    TYPE118 TCPIPSTATISTICS   ; default = notcpipstatistics
    TYPE119 TCPIPSTATISTICS   ; default = notcpipstatistics
          NOTCPINIT           ; default = notcpinit
          NOTCPTERM           ; default = notcpterm
            FTPCLIENT         ; default = noftpclient
            TN3270CLIENT      ; default = notn3270client
            IFSTATISTICS      ; default = noifstatistics
            PORTSTATISTICS    ; default = noportstatistics
            TCPSTACK          ; default = notcpstack
          NOUDPTERM           ; default = noudpterm
;
```

Figure C-5 SC30 TCPIPB profile (PROFB30A)

```
; ---------------------------------------------------------------------
 AUTOLOG 5

   OMPB

 ENDAUTOLOG
; ---------------------------------------------------------------------

PORT
     7 UDP MISCSRVB            ; Miscellaneous Server - echo
     7 TCP MISCSRVB            ; Miscellaneous Server - echo
     9 UDP MISCSRVB            ; Miscellaneous Server - discard
     9 TCP MISCSRVB            ; Miscellaneous Server - discard
    19 UDP MISCSRVB            ; Miscellaneous Server - chargen
    19 TCP MISCSRVB            ; Miscellaneous Server - chargen
    20 TCP OMVS    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.1.8.11     ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server
    25 TCP SMTPB                         ; SMTP Server
    53 TCP NAMED9B                       ; Domain Name Server
    53 UDP NAMED9B                       ; Domain Name Server
   111 TCP PORTMAPB                      ; Portmap Server
   111 UDP PORTMAPB                      ; Portmap Server
;  123 UDP SNTPD               ; Simple Network Time Protocol Server
;  135 UDP LLBD                ; NCS Location Broker
   161 UDP SNMPDB                        ; SNMP Agent
   162 UDP SNMPQEB                       ; SNMP Query Engine
;  389 TCP LDAPSRV             ; LDAP Server
;  443 TCP HTTPS               ; http protocol over TLS/SSL
;  443 UDP HTTPS               ; http protocol over TLS/SSL
   512 TCP OMVS BIND 10.1.8.12        ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.1.8.12        ; UNIX RSHD   D-VIPA
   512 TCP RXSERVEB                      ; TSO  REXECD
   514 TCP RXSERVEB                      ; TSO  RSHD
   515 TCP LPSERVEB                      ; LPD Server
   520 UDP OMPB   NOAUTOLOG              ; OMPROUTE
;  580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERBB                      ; Kerberos
   750 UDP MVSKERBB                      ; Kerberos
   751 TCP ADM@SRVB                      ; Kerberos Admin Server
   751 UDP ADM@SRVB                      ; Kerberos Admin Server
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
```

Figure C-6 SC30 TCPIPB profile (PROFB30A) - continued

Archived

```
;-----------------------------------------------------------------
;Hipersocks: trle definitions are dynamically created on vtam  -
;-----------------------------------------------------------------
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4L IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5L IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6L IPAQIDIO      IUTIQDF6
;
;-----------------------------------------------------------------
; Static VIPA definitions                                        -
;-----------------------------------------------------------------
 DEVICE VIPA1    VIRTUAL 0
 LINK   VIPA1L   VIRTUAL 0 VIPA1
;
;-----------------------------------------------------------------
;Osa definitions... relates to sys1.vtamlst trle definitions   -
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0              -
;-----------------------------------------------------------------
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080L IPAQENET      OSA2080 VLANID 10
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0L IPAQENET      OSA20A0 VLANID 10
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0L IPAQENET      OSA20C0 VLANID 11
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0L IPAQENET      OSA20E0 VLANID 11
;
HOME
   10.1.4.11   IUTIQDF4L
   10.1.5.11   IUTIQDF5L
   10.1.6.11   IUTIQDF6L
   10.1.1.10     VIPA1L
   10.1.2.11    OSA2080L
   10.1.2.12    OSA20A0L
   10.1.3.11    OSA20C0L
   10.1.3.12    OSA20E0L
;
  PRIMARYINTERFACE VIPA1L

 VIPADYNAMIC
  ;-------------------------------------------------------------------
  ;  Set aside 14 addresses for use with BIND and SIOCSVIPA IOCTL    -
  ;              (10.1.8.11 thru .254)                            -
  ;-------------------------------------------------------------------
   VIPARANGE  DEFINE  255.255.255.0  10.1.9.0

 ENDVIPADYNAMIC
```

Figure C-7 SC30 TCPIPB profile (PROFB30A) - continued

Archived

```
ITRACE OFF

START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
START OSA2080
START OSA20A0
START OSA20C0
START OSA20E0
;
******************************** Bottom of Data ********************************
```

Figure C-8 SC30 TCPIPB profile (PROFB30A) - end

SC30 OMPROUTE PROC for TN3270 standalone started task

Figure C-9 shows the JCL procedure for OMPROUTE.

```
***************************** Top of Data ********************
//OMPB   PROC STDENV=OMPENB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
*************************** Bottom of Data ******************
```

Figure C-9 SC30 OMPROUTE PROC (OMPB)

SC30 OMPROUTE STDENV file for TN3270 Standalone Task Scenario

Figure C-10 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPENB30) - 01.04          Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
********************************** Top of Data **********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB30)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB30)'
OMPROUTE_DEBUG_FILE=/etc/omproute/debug30b
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure C-10 SC30 OMPROUTE STDENV file (OMPENB30)

SC30 OMPROUTE CONFIG for TN3270 standalone started task

Figure C-11 shows the config for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB30) - 01.00           Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************** Top of Data ***********************************
Area Area_Number=0.0.0.2
     Stub_Area=YES
     Authentication_type=None
     Import_Summaries=No;
;
OSPF RouterID=10.1.1.10;
Global_Options Ignore_Undefined_Interfaces=yes;
;
Routesa_Config Enabled=No;
;
; hipersocks  IUTIQDF4L
 ospf_interface ip_address=10.1.4.11
                subnet_mask=255.255.255.0
                name=IUTIQDF4L
                attaches_to_area=0.0.0.2
                cost0=10;
;
; hipersocks  IUTIQDF5L
 ospf_interface ip_address=10.1.5.11
                subnet_mask=255.255.255.0
                name=IUTIQDF5L
                ROUTER_PRIORITY=0
                attaches_to_area=0.0.0.2
                cost0=10;
;
; hipersocks  IUTIQDF6L
 ospf_interface ip_address=10.1.6.11
                subnet_mask=255.255.255.0
                name=IUTIQDF6L
                attaches_to_area=0.0.0.2
                cost0=10;
;
; ************************************************************
; *Static VIPA requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; ************************************************************
ospf_interface ip_address=10.1.1.*
           subnet_mask=255.255.255.0
           destination_addr=10.1.1.10
           name=VIPA1L
           Advertise_VIPA_Routes=HOST_ONLY
           Router_Priority=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
```

Figure C-11 SC30 OMPROUTE config (OMPB30)

Archived

```
; OSA Qdio
ospf_interface ip_address=10.1.2.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
;
; OSA Qdio OSA2OCOL
ospf_interface ip_address=10.1.3.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
;
; ************************************************************
; *Dynamic VIPA Range requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.8.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.8.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.1.8.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.2
     Router_Priority=0;
OSPF_INTERFACE
     IP_Address=10.1.9.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.2
     Router_Priority=0;
```

Figure C-12 SC30 OMPROUTE config (OMPB30) - continued

Archived

```
;
; ************************************************************
; *XCF Interfaces
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to just code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
INTERFACE
     IP_Address=10.1.7.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure C-13 SC30 OMPROUTE Config (OMPB30) - end

Archived

Archived

D

Configuration files: Multiple TN3270E Telnet servers and SD scenario

This appendix contains the configuration files used for the multiple TN3270E Telnet servers with the sysplex distribution scenario. Sysplex distribution is abbreviated as SD.

There are two sections in this appendix:

- SC30 files for multiple TN3270E Telnet servers and SD scenario
- SC31 files for multiple TN3270E Telnet servers and SD scenario

© Copyright IBM Corp. 2008. All rights reserved.

SC30 files for multiple TN3270E Telnet servers and SD scenario

The following sections are included:

- SC30 TN3270B Server PROC for multiple TN3270E Telnet servers and SD
- SC30 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD
- SC30 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD
- SC30 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD
- SC30 OMPROUTE PROC for multiple TN3270E Telnet servers and SD
- SC30 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD
- SC30 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD

SC30 TN3270B Server PROC for multiple TN3270E Telnet servers and SD

Figure D-1 shows the PROC JCL for the TN3270E Telnet server.

```
 BROWSE    SYS1.PROCLIB(TN3270B) - 01.02             Line 00000000 Co
 Command ===>                                                  Scroll
********************************** Top of Data ************************
//TN3270B  PROC PARMS='TRC=TN',
//             PROFILE=TELNB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//TN3270B  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
******************************** Bottom of Data *********************
```

Figure D-1 SC30 TN3270B PROC JCL

We started the server task for this scenario by issuing the following command:

```
S TN3270B,PROFILE=TELNB30C
```

SC30 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD

The profile for the TN3270E Telnet server for this scenario is TELNB30C. The naming convention for the profiles is *XXXXyccs*:

- *XXXX* is TELN.
- *y* is B for the stack that was used for the examples in this book.
- *cc* is the &SYSCLONE value of this system.
- *s* is a letter representing the specific scenario, (A, B, C, D).

```
 BROWSE    TCPIPB.TCPPARMS(TELNB30C) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
********************************** Top of Data ***********************************
; ===SC30============ TN3270E Telnet server Profile for Standalone Task =======-
;                                                                              -
;    SSL security. And Sysplex Distribution to this server.                    -
;                                                                              -
; ----------------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
; ----------------------------------------------------------------------------
; These default device type settings will be used by all ports if no   -
; TELNETPARMS or PARMSGROUP is used to override the settings.          -
; They are logmode names shipped in ISTINCDT with the latest level of  -
; VTAM.                                                                -
; ----------------------------------------------------------------------------
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
    ;
ENDTELNETGLOBALS
;
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
```

Figure D-2 SC30 TN3270B profile (TELNB30C)

Archived

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC30BB01..SC30BB99
  ENDDEFAULTLUS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL SC30N*          ; Netview
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING SAF TCPIP/SharedRing1     ;keyring used by all secure
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
ENDTELNETPARMS
;
BEGINVTAM
  PORT 992
  DEFAULTLUS
      SC30BS01..SC30BS99
  ENDDEFAULTLUS

; ---------------------------------------------------------------------
; This NOSSL        group is mapped to use no SSL security.           -
; NOLUSESSIONPEND = Terminate connection upon a logoff from the appl   -
; ---------------------------------------------------------------------
  PARMSGROUP NOSSL
   NOLUSESSIONPEND
     CONNTYPE BASIC  ; support non-secure, overrides telnetparms
  ENDPARMSGROUP

; ---------------------------------------------------------------------
; The SSLPLAIN group is mapped to use SSL security                      -
;                 with no Client Authentication required              -
; LUSESSIONPEND = Force a requeue back to the USS table upon logoff    -
; ---------------------------------------------------------------------
  PARMSGROUP SSLPLAIN
     LUSESSIONPEND
     CONNTYPE SECURE ; says plain SSL, no client auth specified
  ENDPARMSGROUP      ; and negotiate all available encryption algorithms
```

Figure D-3 SC30 TN3270B profile (TELNB30C) - continued

```
; ---------------------------------------------------------------------
; The SSLCERTS group is mapped to use SSL security                    -
;                  and to require Client Authentication (certificates) -
; NOLUSESSIONPEND = Terminate connection upon a logoff from the appl   -
; ---------------------------------------------------------------------
  PARMSGROUP SSLCERTS
     NOLUSESSIONPEND
     CONNTYPE SECURE       ; Support SSL
     CLIENTAUTH SSLCERT    ; Client Certificate required
     ENCRYPT SSL_DES_SHA   ; use these only, do not consider any others
             SSL_3DES_SHA
     ENDENCRYPT
  ENDPARMSGROUP

; ---------------------------------------------------------------------
; The UP2USER      group is mapped to use ANY security (user's choice) -
;                  with no Client Authentication (no certificates)     -
; LUSESSIONPEND = Force a requeue back to the defaultappl upon logoff  -
; ---------------------------------------------------------------------
  PARMSGROUP UP2USER
     LUSESSIONPEND
     CONNTYPE ANY          ; Whatever User wants to do
  ENDPARMSGROUP

  DESTIPGROUP GENERALUSER 10.1.8.11 ENDDESTIPGROUP
  DESTIPGROUP ADMIN       10.1.8.12 ENDDESTIPGROUP
  DESTIPGROUP PAYROLL     10.1.8.13 ENDDESTIPGROUP
  DESTIPGROUP SHIPPING    10.1.1.10 ENDDESTIPGROUP
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.2   ENDDESTIPGROUP

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLPO  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER      DESTIPGRP,SHIPPING
  DEFAULTAPPL TSO       DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE
 ;-------------------------------------------------------------------
 ; There is no DEFAULTAPPL nor USSTCB coded as a default catch all -
 ; So, if any user connects using any other IP address than the    -
 ; four defined by the DESTIPGROUPs above, the Network Solicitor   -
 ; prompt panel will be displayed to that user, in BASIC mode.     -
 ;-------------------------------------------------------------------
  ALLOWAPPL SC30N*          ; Netview
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
****************************** Bottom of Data ******************************
```

Figure D-4 SC30 TN3270B profile (TELNB30C) - end

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC30B*?nn*):

- *XXXX* is the &SYSNAME of the system.
- *y* is B for stack B that was used for all of the examples in this book.
- *?* is B for the Basic port (23) and S for the Secure port (992).
- *nn* is 01 through 99.

Because TN3270 maintains a master list of LU names in use across all ports, we could have decided to use the same group of default LU names for all ports in our setup. We chose different pool names, one for basic and one for secure, to help us verify which port we are connected to in our display examples.

SC30 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD

The PROC JCL for the stack is TCPIPB.

```
//TCPIPB    PROC PARMS='CTRACE(CTIEZB00),IDS=00',
//               PROFILE=PROFB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE
//TCPIPB    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//               PARM=('&PARMS',
//         'ENVAR("RESOLVER_CONFIG=//''TCPIPB.TCPPARMS(&TCPDATA)'''")')
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*TNDBCSER DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure D-5 SC30 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB30C
```

SC30 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD

The PROFILE for the stack is PROFB30C.

```
 BROWSE    TCPIPB.TCPPARMS(PROFB30C) - 01.10          Line 00000000 Col 001 080
 Command ===>                                                   Scroll ===> CSR
********************************* Top of Data **********************************
;
; TCPIP.PROFILE.TCPIP
; ===================
;
GLOBALCONFIG
   ECSALIMIT 0M              ; default = 0M
   POOLLIMIT 0M              ; default = 0M
   TCPIPSTATISTICS           ; default = notcpipstatistics
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
   DATAGRAMFWD
   SYSPLEXROUTING
   DYNAMICXCF 10.1.7.11 255.255.255.0 8
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
   RESTRICTLOWPORTS          ; default = restrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   DELAYACKS                 ; default = delayacks
;
UDPCONFIG
   RESTRICTLOWPORTS          ; default = restrictlowports
   UDPCHKSUM                 ; default = udpchksum
   UDPSENDBFRSIZE   65535    ; default = 64K
   UDPRCVBUFRSIZE   65535    ; default = 64K
   NOUDPQUEUELIMIT             ; default = udpqueuelimit
;
 SMFCONFIG                    ;
    TYPE118 TCPIPSTATISTICS   ; default = notcpipstatistics
    TYPE119 TCPIPSTATISTICS   ; default = notcpipstatistics
          NOTCPINIT           ; default = notcpinit
          NOTCPTERM           ; default = notcpterm
            FTPCLIENT         ; default = noftpclient
            TN3270CLIENT      ; default = notn3270client
            IFSTATISTICS      ; default = noifstatistics
            PORTSTATISTICS    ; default = noportstatistics
            TCPSTACK          ; default = notcpstack
          NOUDPTERM           ; default = noudpterm
```

Figure D-6 SC30 TCPIPB profile (PROFB30C) - continued

Archived

```
; ---------------------------------------------------------------------
 AUTOLOG 5

   OMPB

 ENDAUTOLOG
; ---------------------------------------------------------------------

PORT
     7 UDP MISCSRVB            ; Miscellaneous Server - echo
     7 TCP MISCSRVB            ; Miscellaneous Server - echo
     9 UDP MISCSRVB            ; Miscellaneous Server - discard
     9 TCP MISCSRVB            ; Miscellaneous Server - discard
    19 UDP MISCSRVB            ; Miscellaneous Server - chargen
    19 TCP MISCSRVB            ; Miscellaneous Server - chargen
    20 TCP OMVS    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.1.9.11 ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server
    25 TCP SMTPB                         ; SMTP Server
    53 TCP NAMED9B                       ; Domain Name Server
    53 UDP NAMED9B                       ; Domain Name Server
   111 TCP PORTMAPB                      ; Portmap Server
   111 UDP PORTMAPB                      ; Portmap Server
;  123 UDP SNTPD               ; Simple Network Time Protocol Server
;  135 UDP LLBD                ; NCS Location Broker
   161 UDP SNMPDB                        ; SNMP Agent
   162 UDP SNMPQEB                       ; SNMP Query Engine
;  389 TCP LDAPSRV             ; LDAP Server
;  443 TCP HTTPS               ; http protocol over TLS/SSL
;  443 UDP HTTPS               ; http protocol over TLS/SSL
   512 TCP OMVS BIND 10.1.9.12 ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.1.9.12 ; UNIX RSHD   D-VIPA
   512 TCP RXSERVEB                      ; TSO  REXECD
   514 TCP RXSERVEB                      ; TSO  RSHD
   515 TCP LPSERVEB                      ; LPD Server
   520 UDP OMPB    NOAUTOLOG             ; OMPROUTE
;  580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERBB                      ; Kerberos
   750 UDP MVSKERBB                      ; Kerberos
   751 TCP ADM@SRVB                      ; Kerberos Admin Server
   751 UDP ADM@SRVB                      ; Kerberos Admin Server
   992 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server SECURE
   12000 UDP NET                 ; Enterprise Extender signal data
   12001 UDP NET                 ; Enterprise Extender NETWORK priority
   12002 UDP NET                 ; Enterprise Extender HIGH priority
   12003 UDP NET                 ; Enterprise Extender MEDIUM priority
   12004 UDP NET                 ; Enterprise Extender LOW priority
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
```

Figure D-7 SC30 TCPIPB profile (PROFB30C) - continued

Archived

```
;-----------------------------------------------------------------
;Hipersocks: trle definitions are dynamically created on vtam  -
;-----------------------------------------------------------------
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4L  IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5L  IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6L  IPAQIDIO      IUTIQDF6
;
;-----------------------------------------------------------------
; Static VIPA definitions                                        -
;-----------------------------------------------------------------
 DEVICE VIPA1    VIRTUAL 0
 LINK   VIPA1L   VIRTUAL 0 VIPA1
;
;-----------------------------------------------------------------
;Osa definitions... relates to sys1.vtamlst trle definitions   -
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0              -
;-----------------------------------------------------------------
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080L  IPAQENET      OSA2080 VLANID 10
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0L  IPAQENET      OSA20A0 VLANID 10
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0L  IPAQENET      OSA20C0 VLANID 11
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0L  IPAQENET      OSA20E0 VLANID 11
;
HOME
   10.1.4.11 IUTIQDF4L
   10.1.5.11 IUTIQDF5L
   10.1.6.11 IUTIQDF6L
   10.1.1.10 VIPA1L
   10.1.2.11 OSA2080L
   10.1.2.12 OSA20A0L
   10.1.3.11 OSA20C0L
   10.1.3.12 OSA20E0L
;
  PRIMARYINTERFACE VIPA1L
```

Figure D-8 SC30 TCPIPB profile (PROFB30C) - continued

Archived

```
VIPADYNAMIC
 ;-------------------------------------------------------------------
 ;  Set aside addresses for use with BIND and SIOCSVIPA IOCTL    -
 ;-------------------------------------------------------------------
  VIPARANGE  DEFINE  255.255.255.0 10.1.9.0

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
 ;-------------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.25 ;FTP
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                        10.1.8.25 PORT 20 21
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
 ;-------------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21 ;TN3270 General
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                        10.1.8.21 PORT 23  992
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
 ;-------------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22  ;TN3270 Admin
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                        10.1.8.22 PORT 23  992
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
 ;-------------------------------------------------------------------
  VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23  ;TN3270 Payrol
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                        10.1.8.23   PORT 23  992
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;      Distribute to 10.1.7.11 via normal XCF   (no viparoute)  -
 ;      Distribute to 10.1.7.21 via IP routing   (   viparoute)  -
 ;-------------------------------------------------------------------
;;VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa
 VIPAROUTE      DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa
ENDVIPADYNAMIC
```

Figure D-9 SC30 TCPIPB profile (PROFB30C) - continued

```
ITRACE OFF

START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
START OSA2080
START OSA20A0
START OSA20C0
START OSA20E0
;
******************************** Bottom of Data ********************************
```

Figure D-10 SC30 TCPIPB profile (PROFB30C) - end

SC30 OMPROUTE PROC for multiple TN3270E Telnet servers and SD

Figure D-11 shows the PROC JCL for OMPROUTE.

```
***************************** Top of Data ********************
//OMPB   PROC STDENV=OMPENB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
*************************** Bottom of Data ******************
```

Figure D-11 SC30 OMPROUTE PROC (OMPB)

SC30 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD

Figure D-12 shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPENB30) - 01.04         Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
********************************** Top of Data **********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB30)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB30)'
OMPROUTE_DEBUG_FILE=/etc/omproute/debug30b
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure D-12 SC30 OMPROUTE STDENV file (OMPENB30)

SC30 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD

Figure D-13 on page 354 shows the config for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB30) - 01.00           Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************** Top of Data ***********************************
Area Area_Number=0.0.0.2
     Stub_Area=YES
     Authentication_type=None
     Import_Summaries=No;
;
OSPF RouterID=10.1.1.10;
Global_Options Ignore_Undefined_Interfaces=yes;
;
Routesa_Config Enabled=No;
;
; hipersocks  IUTIQDF4L
 ospf_interface ip_address=10.1.4.11
                subnet_mask=255.255.255.0
                name=IUTIQDF4L
                attaches_to_area=0.0.0.2
                cost0=10;
;
; hipersocks  IUTIQDF5L
 ospf_interface ip_address=10.1.5.11
                subnet_mask=255.255.255.0
                name=IUTIQDF5L
                ROUTER_PRIORITY=0
                attaches_to_area=0.0.0.2
                cost0=10;
;
; hipersocks  IUTIQDF6L
 ospf_interface ip_address=10.1.6.11
                subnet_mask=255.255.255.0
                name=IUTIQDF6L
                attaches_to_area=0.0.0.2
                cost0=10;
;
; ************************************************************
; *Static VIPA requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; ************************************************************
ospf_interface ip_address=10.1.1.10
           subnet_mask=255.255.255.252
           destination_addr=10.1.1.10
           name=VIPA1L
           Advertise_VIPA_Routes=HOST_ONLY
           Router_Priority=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
```

Figure D-13 SC30 OMPROUTE Config (OMPB30) - continued

Archived

```
; OSA Qdio
ospf_interface ip_address=10.1.2.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
;

ospf_interface ip_address=10.1.3.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=10
          mtu=1500;
;
;
; ************************************************************
; *Dynamic VIPA Range requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.8.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.8.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.1.8.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.2
     Router_Priority=0;
OSPF_INTERFACE
     IP_Address=10.1.9.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.2
     Router_Priority=0;
```

Figure D-14 SC30 OMPROUTE Config (OMPB30) - continued

Archived

```
;
; ************************************************************
; *XCF Interfaces
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
INTERFACE
     IP_Address=10.1.7.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure D-15 SC30 OMPROUTE Config (OMPB30) - end

SC31 files for multiple TN3270E Telnet servers and SD scenario

The following sections are included below:

- SC31 TN3270B Server PROC for multiple TN3270E Telnet servers and SD
- SC31 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD
- SC31 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD
- SC31 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD
- SC31 OMPROUTE PROC for multiple TN3270E Telnet servers and SD
- SC31 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD
- SC31 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD

SC31 TN3270B Server PROC for multiple TN3270E Telnet servers and SD

Figure D-16 on page 357 shows the PROC JCL for the TN3270E Telnet server.

```
 BROWSE    SYS1.PROCLIB(TN3270B) - 01.02             Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************** Top of Data ***********************************
//TN3270B  PROC PARMS='TRC=TN',
//             PROFILE=TELNB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE.
//*
//*  TRC=TN indicates to use CTRACE(CTIEZBTN) parmlib control member
//*
//TN3270B  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//STEPLIB  DD DISP=SHR,DSN=SYS1.LOCAL.VTAMLIB
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&TCPDATA.)
```

Figure D-16 SC31 TN3270B PROC JCL

We started the server task for this scenario by issuing the following command:

```
S TN3270B,PROFILE=TELNB31C
```

SC31 TN3270B Server PROFILE for multiple TN3270E Telnet servers and SD

The profile for the TN3270E Telnet server for this scenario is TELNB31C. The naming convention for the profiles is *XXXXyccs*:

- *XXXX* is TELN.
- *y* is B for the stack that was used for the examples in this book.
- *cc* is the &SYSCLONE value of this system.
- *s* is a letter representing the specific scenario, (A, B, C, D).

Archived

```
 BROWSE    TCPIPB.TCPPARMS(TELNB31C) - 01.03         Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
********************************** Top of Data ***********************************
; ===SC31============ TN3270E Telnet server Profile for Standalone Task =======-
;                                                                              -
;    SSL security. And Sysplex Distribution to this server.                    -
;                                                                              -
; ----------------------------------------------------------------------------
;
TELNETGLOBALS
    TCPIPJOBNAME TCPIPB
; ----------------------------------------------------------------------------
; These default device type settings will be used by all ports if no   -
; TELNETPARMS or PARMSGROUP is used to override the settings.          -
; They are logmode names shipped in ISTINCDT with the latest level of  -
; VTAM.                                                                -
; ----------------------------------------------------------------------------
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
    ;
ENDTELNETGLOBALS
;
TELNETPARMS
    PORT 23
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
```

Figure D-17 SC31 TN3270B profile (TELNB31C) - continued

```
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC31BB01..SC31BB99
  ENDDEFAULTLUS

  DEFAULTAPPL TSO           ; All users go to TSO
  ALLOWAPPL SC30N*          ; Netview
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
TELNETPARMS
    SECUREPORT 992  ;Port 992 will support SSL
    KEYRING SAF TCPIP/SharedRing1     ;keyring used by all secure
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
ENDTELNETPARMS
;
BEGINVTAM
  PORT 992
  DEFAULTLUS
      SC31BS01..SC31BS99
  ENDDEFAULTLUS

; ---------------------------------------------------------------------
; This NOSSL        group is mapped to use no SSL security.           -
; NOLUSESSIONPEND = Terminate connection upon a logoff from the appl   -
; ---------------------------------------------------------------------
  PARMSGROUP NOSSL
   NOLUSESSIONPEND
     CONNTYPE BASIC  ; support non-secure, overrides telnetparms
  ENDPARMSGROUP

; ---------------------------------------------------------------------
; The SSLPLAIN group is mapped to use SSL security                      -
;                 with no Client Authentication required              -
; LUSESSIONPEND = Force a requeue back to the USS table upon logoff    -
; ---------------------------------------------------------------------
  PARMSGROUP SSLPLAIN
     LUSESSIONPEND
     CONNTYPE SECURE ; says plain SSL, no client auth specified
  ENDPARMSGROUP      ; and negotiate all available encryption algorithms
```

Figure D-18 SC31 TN3270B profile (TELNB31C) - continued

```
; ---------------------------------------------------------------------
; The SSLCERTS group is mapped to use SSL security                    -
;                  and to require Client Authentication (certificates) -
; NOLUSESSIONPEND = Terminate connection upon a logoff from the appl   -
; ---------------------------------------------------------------------
  PARMSGROUP SSLCERTS
   NOLUSESSIONPEND
     CONNTYPE SECURE       ; Support SSL
     CLIENTAUTH SSLCERT    ; Client Certificate required
     ENCRYPT SSL_DES_SHA   ; use these only, do not consider any others
             SSL_3DES_SHA
     ENDENCRYPT
  ENDPARMSGROUP

; ---------------------------------------------------------------------
; The UP2USER      group is mapped to use ANY security (user's choice) -
;                  with no Client Authentication (no certificates)     -
; LUSESSIONPEND = Force a requeue back to the defaultappl upon logoff  -
; ---------------------------------------------------------------------
  PARMSGROUP UP2USER
     LUSESSIONPEND
     CONNTYPE ANY          ; Whatever User wants to do
  ENDPARMSGROUP

  DESTIPGROUP GENERALUSER 10.1.8.21   ENDDESTIPGROUP
  DESTIPGROUP ADMIN       10.1.8.22   ENDDESTIPGROUP
  DESTIPGROUP PAYROLL     10.1.8.23   ENDDESTIPGROUP
  DESTIPGROUP SHIPPING    10.1.1.20   ENDDESTIPGROUP
  DESTIPGROUP ANY1ELSE    255.0.0.0:10.0.0.2   ENDDESTIPGROUP

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER
  DEFAULTAPPL SC30N     DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL
  DEFAULTAPPL CICSCLPO  DESTIPGRP,PAYROLL

  PARMSMAP UP2USER      DESTIPGRP,SHIPPING
  DEFAULTAPPL TSO       DESTIPGRP,SHIPPING

  PARMSMAP NOSSL        DESTIPGRP,ANY1ELSE
 ;-------------------------------------------------------------------
 ; There is no DEFAULTAPPL nor USSTCB coded as a default catch all -
 ; So, if any user connects using any other IP address than the    -
 ; four defined by the DESTIPGROUPs above, the Network Solicitor   -
 ; prompt panel will be displayed to that user, in BASIC mode.     -
 ;-------------------------------------------------------------------
  ALLOWAPPL SC30N*          ; Netview
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
****************************** Bottom of Data ******************************
```

Figure D-19 SC31 TN3270B profile (TELNB31C) - end

The naming convention used for the DEFAULTLUS is *XXXXyznn*: (SC30B*?nn*):

- *XXXX* is the &SYSNAME of the system.
- *y* is B for stack B that was used for all of the examples in this book.
- *?* is B for the basic port (23) and S for the secure port (992).
- *nn* is 01 through 99.

Because TN3270 maintains a master list of LU names in use across all ports, we could have decided to use the same group of default LU names for all ports in our setup. We chose different pool names, one for basic and one for secure, to help us verify which port we are connected to in our display examples.

SC31 TCPIPB stack PROC for multiple TN3270E Telnet servers and SD

The PROC JCL for the stack is TCPIPB.

```
//TCPIPB    PROC PARMS='CTRACE(CTIEZB00),IDS=00',
//             PROFILE=PROFB&SYSCLONE.,TCPDATA=DATAB&SYSCLONE
//TCPIPB    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM=('&PARMS',
//         'ENVAR("RESOLVER_CONFIG=//''TCPIPB.TCPPARMS(&TCPDATA)'''")')
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD SYSOUT=*
//*TNDBCSCN DD DSN=TCPIP.SEZAINST(TNDBCSCN),DISP=SHR
//*TNDBCSXL DD DSN=TCPIP.SEZAXLD2,DISP=SHR
//*TNDBCSER DD SYSOUT=*
//PROFILE  DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&PROFILE.)
//SYSTCPD  DD DSN=TCPIPB.TCPPARMS(&TCPDATA.),DISP=SHR
```

Figure D-20 SC31 TCPIPB PROC JCL

We started the TCPIPB started task for this scenario by issuing the following command:

```
S TCPIPB,PROFILE=PROFB31C
```

SC31 TCPIPB stack PROFILE for multiple TN3270E Telnet servers and SD

The PROFILE for the stack is PROFB31C.

```
 BROWSE    TCPIPB.TCPPARMS(PROFB31C) - 01.09         Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
********************************* Top of Data **********************************
;
; TCPIP.PROFILE.TCPIP   for SC31: Sysplex distributing TN3270
; ===================
;
GLOBALCONFIG
   ECSALIMIT 0M              ; default = 0M
   POOLLIMIT 0M              ; default = 0M
   TCPIPSTATISTICS           ; default = notcpipstatistics
   SYSPLEXMONITOR DELAYJOIN NORECOVERY TIMERSECS 60
;
IPCONFIG
   ARPTO      1200
   SOURCEVIPA
   IGNOREREDIRECT
   MULTIPATH PERCONNECTION
   PATHMTUDISCOVERY
   DATAGRAMFWD
   SYSPLEXROUTING
   DYNAMICXCF 10.1.7.21 255.255.255.0 8
;
; possible bufrsizes  16K  32K  64K  128K  256K  512K
TCPCONFIG
   FINWAIT2TIME 600          ; in seconds,  default = 600
   INTERVAL    15            ; in minutes,  default = 120
   RESTRICTLOWPORTS          ; default = restrictlowports
   SENDGARBAGE FALSE         ; default = false
   TCPSENDBFRSIZE  256K      ; default = 16K
   TCPRCVBUFRSIZE  256K      ; default = 16K
   TCPMAXRCVBUFRSIZE  512K   ; default = 256K
   TCPTIMESTAMP              ; default = tcptimestamp
   DELAYACKS                 ; default = delayacks
;
UDPCONFIG
   RESTRICTLOWPORTS          ; default = restrictlowports
   UDPCHKSUM                 ; default = udpchksum
   UDPSENDBFRSIZE   65535    ; default = 64K
   UDPRCVBUFRSIZE   65535    ; default = 64K
 NOUDPQUEUELIMIT             ; default = udpqueuelimit
;
 SMFCONFIG                    ;
    TYPE118 TCPIPSTATISTICS   ; default = notcpipstatistics
    TYPE119 TCPIPSTATISTICS   ; default = notcpipstatistics
          NOTCPINIT           ; default = notcpinit
          NOTCPTERM           ; default = notcpterm
            FTPCLIENT         ; default = noftpclient
            TN3270CLIENT      ; default = notn3270client
            IFSTATISTICS      ; default = noifstatistics
            PORTSTATISTICS    ; default = noportstatistics
            TCPSTACK          ; default = notcpstack
          NOUDPTERM           ; default = noudpterm
```

Figure D-21 SC31 TCPIPB profile (PROFB31C) - continued

Archived

```
; ---------------------------------------------------------------------
 AUTOLOG 5

   OMPB

 ENDAUTOLOG
; ---------------------------------------------------------------------

PORT
     7 UDP MISCSRVB            ; Miscellaneous Server - echo
     7 TCP MISCSRVB            ; Miscellaneous Server - echo
     9 UDP MISCSRVB            ; Miscellaneous Server - discard
     9 TCP MISCSRVB            ; Miscellaneous Server - discard
    19 UDP MISCSRVB            ; Miscellaneous Server - chargen
    19 TCP MISCSRVB            ; Miscellaneous Server - chargen
    20 TCP OMVS    NOAUTOLOG NODELAYACKS ; FTP Server
    21 TCP OMVS                          ; control port
    23 TCP OMVS    BIND 10.1.9.21 ; OE Telnet Server D-VIPA
    23 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server
    25 TCP SMTPB                         ; SMTP Server
    53 TCP NAMED9B                       ; Domain Name Server
    53 UDP NAMED9B                       ; Domain Name Server
   111 TCP PORTMAPB                      ; Portmap Server
   111 UDP PORTMAPB                      ; Portmap Server
;  123 UDP SNTPD               ; Simple Network Time Protocol Server
;  135 UDP LLBD                ; NCS Location Broker
   161 UDP SNMPDB                        ; SNMP Agent
   162 UDP SNMPQEB                       ; SNMP Query Engine
;  389 TCP LDAPSRV             ; LDAP Server
;  443 TCP HTTPS               ; http protocol over TLS/SSL
;  443 UDP HTTPS               ; http protocol over TLS/SSL
   512 TCP OMVS BIND 10.1.9.22 ; UNIX REXECD D-VIPA
   514 TCP OMVS BIND 10.1.9.22 ; UNIX RSHD   D-VIPA
   512 TCP RXSERVEB                      ; TSO  REXECD
   514 TCP RXSERVEB                      ; TSO  RSHD
   515 TCP LPSERVEB                      ; LPD Server
   520 UDP OMPB    NOAUTOLOG             ; OMPROUTE
;  580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERBB                      ; Kerberos
   750 UDP MVSKERBB                      ; Kerberos
   751 TCP ADM@SRVB                      ; Kerberos Admin Server
   751 UDP ADM@SRVB                      ; Kerberos Admin Server
   992 TCP TN3270B NOAUTOLOG             ; MVS Telnet Server SECURE
   12000 UDP NET                 ; Enterprise Extender signal data
   12001 UDP NET                 ; Enterprise Extender NETWORK priority
   12002 UDP NET                 ; Enterprise Extender HIGH priority
   12003 UDP NET                 ; Enterprise Extender MEDIUM priority
   12004 UDP NET                 ; Enterprise Extender LOW priority
;
; Used for Netview      - needed for AON
; SACONFIG ENABLED COMMUNITY public AGENT 161
SACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
```

Figure D-22 SC31 TCPIPB profile (PROFB31C) - continued

Archived

```
;-----------------------------------------------------------------
;Hipersocks: trle definitions are dynamically created on vtam  -
;-----------------------------------------------------------------
 DEVICE IUTIQDF4 MPCIPA
 LINK   IUTIQDF4L  IPAQIDIO      IUTIQDF4
 DEVICE IUTIQDF5 MPCIPA
 LINK   IUTIQDF5L  IPAQIDIO      IUTIQDF5
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6L  IPAQIDIO      IUTIQDF6
;
;-----------------------------------------------------------------
; Static VIPA definitions                                        -
;-----------------------------------------------------------------
 DEVICE VIPA1    VIRTUAL 0
 LINK   VIPA1L VIRTUAL 0     VIPA1
;
;-----------------------------------------------------------------
;Osa definitions... relates to sys1.vtamlst trle definitions   -
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0              -
;-----------------------------------------------------------------
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080L  IPAQENET      OSA2080 VLANID 10
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0L  IPAQENET      OSA20A0 VLANID 10
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0L  IPAQENET      OSA20C0 VLANID 11
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0L  IPAQENET      OSA20E0 VLANID 11
;
HOME
   10.1.4.21 IUTIQDF4L
   10.1.5.21 IUTIQDF5L
   10.1.6.21 IUTIQDF6L
   10.1.1.20   VIPA1L
   10.1.2.21 OSA2080L
   10.1.2.22 OSA20A0L
   10.1.3.21 OSA20C0L
   10.1.3.22 OSA20E0L
;
  PRIMARYINTERFACE VIPA1L
```

Figure D-23 SC31 TCPIPB profile (PROFB31C) - continued

Archived

```
VIPADYNAMIC
 ;-------------------------------------------------------------------
 ;  Set aside addresses for use with BIND and SIOCSVIPA IOCTL    -
 ;-------------------------------------------------------------------
  VIPARANGE  DEFINE  255.255.255.0 10.1.9.0

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for FTP using BASEWLM algorithm     -
 ;-------------------------------------------------------------------
  VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.25   ;FTP
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                        10.1.8.25    PORT 20 21
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using ROUNDROBIN         -
 ;-------------------------------------------------------------------
  VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.21  ;TN3270 General
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD ROUNDROBIN
                        10.1.8.21   PORT 23  992
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using BASEWLM            -
 ;-------------------------------------------------------------------
  VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.22  ;TN3270 Admin
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD BASEWLM
                        10.1.8.22   PORT 23  992
                 DESTIP 10.1.7.11
                        10.1.7.21

;-------------------------------------------------------------------
 ;  Set up Sysplex Distribution for TN3270 using SERVERWLM          -
 ;-------------------------------------------------------------------
  VIPABACKUP 200 MOVE IMMED 255.255.255.0 10.1.8.23  ;TN3270 Payrol
  VIPADISTRIBUTE DEFINE SYSPLEXPORTS DISTMETHOD SERVERWLM
                        10.1.8.23   PORT 23  992
                 DESTIP 10.1.7.11
                        10.1.7.21

 ;-------------------------------------------------------------------
 ;      Distribute to 10.1.7.11 via IP routing   (   viparoute)  -
 ;      Distribute to 10.1.7.21 via normal XCF   (no viparoute)  -
 ;-------------------------------------------------------------------
  VIPAROUTE      DEFINE 10.1.7.11 10.1.1.10 ; sc30's static vipa
;;VIPAROUTE      DEFINE 10.1.7.21 10.1.1.20 ; sc31's static vipa
ENDVIPADYNAMIC
```

Figure D-24 SC31 TCPIPB profile (PROFB31C) - continued

Archived

```
ITRACE OFF

START IUTIQDF4
START IUTIQDF5
START IUTIQDF6
START OSA2080
START OSA20A0
START OSA20C0
START OSA20E0
;
******************************** Bottom of Data ********************************
```

Figure D-25 SC31 TCPIPB profile (PROFB31C) - end

SC31 OMPROUTE PROC for multiple TN3270E Telnet servers and SD

Figure D-26 shows the PROC JCL for OMPROUTE.

```
****************************** Top of Data ********************
//OMPB   PROC STDENV=OMPENB&SYSCLONE
//OMPB   EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//            '"_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD DISP=SHR,DSN=TCPIPB.TCPPARMS(&STDENV)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*STDOUT   DD PATH='/etc/omproute/omproute.stdout',
//*             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
***************************** Bottom of Data ******************
```

Figure D-26 SC31 OMPROUTE PROC (OMPB)

SC31 OMPROUTE STDENV file for multiple TN3270E Telnet servers and SD

This shows the standard environment variable file for OMPROUTE.

```
 BROWSE    TCPIPB.TCPPARMS(OMPENB31) - 01.05         Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
********************************** Top of Data **********************************
RESOLVER_CONFIG=//'TCPIPB.TCPPARMS(DATAB31)'
OMPROUTE_FILE=//'TCPIPB.TCPPARMS(OMPB31)'
OMPROUTE_DEBUG_FILE=/etc/omproute/debug31b
OMPROUTE_DEBUG_FILE_CONTROL=100000,5
OMPROUTE_OPTIONS=hello_hi
******************************** Bottom of Data ********************************
```

Figure D-27 SC31 OMPROUTE STDENV file (OMPENB31)

SC31 OMPROUTE CONFIG for multiple TN3270E Telnet servers and SD

The config for OMPROUTE is OMPB31.

```
 BROWSE    TCPIPB.TCPPARMS(OMPB31) - 01.02            Line 00000000 Col 001 080
 Command ===>                                                Scroll ===> CSR
******************************** Top of Data ***********************************
Area Area_Number=0.0.0.2
     Stub_Area=NO
     Authentication_type=None
     Import_Summaries=No;
;
OSPF RouterID=10.1.1.20;
;
Global_Options Ignore_Undefined_Interfaces=yes;
;
; AS_Boundary_routing
;  Import_Direct_Routes=yes;
;
Routesa_Config Enabled=No;
;
; hipersocks  IUTIQDF4L
 ospf_interface ip_address=10.1.4.21
                subnet_mask=255.255.255.0
                name=IUTIQDF4L
                attaches_to_area=0.0.0.2
                cost0=10;
;
; hipersocks  IUTIQDF5L
 ospf_interface ip_address=10.1.5.21
                subnet_mask=255.255.255.0
                name=IUTIQDF5L
                ROUTER_PRIORITY=0
                attaches_to_area=0.0.0.2
                cost0=10;
;
; hipersocks  IUTIQDF6L
 ospf_interface ip_address=10.1.6.21
                subnet_mask=255.255.255.0
                name=IUTIQDF6L
                attaches_to_area=0.0.0.2
                cost0=10;
;
```

Figure D-28 SC31 OMPROUTE config (OMPB31)

Archived

```
; ************************************************************
; *Static VIPA requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; ************************************************************
ospf_interface ip_address=10.1.1.20
           subnet_mask=255.255.255.252
           destination_addr=10.1.1.20
           name=VIPA1L
           Advertise_VIPA_Routes=HOST_ONLY
           Router_Priority=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
;
; OSA Qdio
ospf_interface ip_address=10.1.2.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
;
; OSA Qdio
ospf_interface ip_address=10.1.3.*
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=1500;
```

Figure D-29 SC31 OMPROUTE config (OMPB31) - continued

Archived

```
;
; ************************************************************
; *Dynamic VIPA Range requirements
; *      Although VIPA interfaces are not participating in the OSPF
; *      protocol, they must be defined as an OSPF_INTERFACE so
; *      they will be advertised to the default router.
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.8.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.8.* subnet should be dedicated to D-VIPA use.
; ************************************************************
OSPF_INTERFACE
     IP_Address=10.1.8.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.2
     Router_Priority=0;
OSPF_INTERFACE
     IP_Address=10.1.9.*
     Subnet_Mask=255.255.255.0
     Advertise_VIPA_Routes=HOST_ONLY
     MTU=1500
     Cost0=10
     Attaches_To_Area=0.0.0.2
     Router_Priority=0;

; ************************************************************
; *XCF Interfaces
; *XCF Interfaces Created dynamically via DYNAMICXCF stmt in TCPIP
; *XCF Interfaces are point-to-multipoint interfaces
; *XCF Interfaces should not be advertised via OSPF, they are internal
; *               to the stack, and therefore external to OSPF
; *Do not specify name for dynamically created interfaces
; *   (* wildcard for ip address is to allow dynamics to work....
; *      however this means that any address in the 10.1.7.*
; *      network that may be found on the stack will be
; *      matched to this interface statement...be cautious....
; *      the 10.1.7.* subnet should be dedicated to XCF use.
; *
; *Another way to accomplish all this is to just code the following stmt:
; *   Global_Options Ignore_Undefined_Interfaces=yes;
; *
; ************************************************************
INTERFACE
     IP_Address=10.1.7.*
     Subnet_Mask=255.255.255.0
     MTU=1500;
;
******************************** Bottom of Data ********************************
```

Figure D-30 SC31 OMPROUTE config (OMPB31) - end

Archived

E

FTP and translation tables

This appendix shows you how to work with translation tables in the FTP environment. It covers how to override the default tables for single-byte character sets (SBCS), double-byte character sets (DBCS), and multi-byte character sets (MBCS). Translation support is well documented in the z/OS Communications Server product publication manuals. This appendix does not duplicate that documentation, but instead shows examples of how to use FTP subcommands to manage the translation process.

This section discusses the following translation topics.

Section	Topic
“Conceptual overview of FTP translation” on page 372	General FTP translation support.
“Using the RFC2389 and RFC2640 FTP features” on page 377	Special translation features.
“Selecting translation tables” on page 379	Examples of using various translation functions.

© Copyright IBM Corp. 2008

Conceptual overview of FTP translation

z/OS Communications Server is delivered with translation tables supporting many different languages, including both single-byte character sets (SBCS) and double-byte character sets (DBCS). The FTP implementation supports the transfer of Unicode data encoded in UCS-2, UTF-8, UTF-16, and other encoding schemes. Unicode is capable of encoding all of the characters of the major scripts used throughout the world.

What is translation?

TCP/IP uses translation tables to convert transmitted data from EBCDIC to ASCII and from ASCII to EBCDIC. TCP/IP provides standard tables that are used as the default if you do not customize your own. When either the z/OS FTP client or the z/OS FTP server is in an FTP session with an ASCII host and you use a data type of ASCII, the data will be translated between EBCDIC and ASCII. This translation takes place on the MVS system.

In addition to single-byte ASCII/EBCDIC conversion, FTP supports DBCS and Unicode conversions to/from EBCDIC.

How does translation work?

The FTP.DATA file is read when ftpd is started and the translate tables are set up in the initial address space which has access to DD statement allocation. A process is then forked to run in the background but an execv() is not done at this stage. Therefore, it is an exact copy of the original address space and has all the configuration data. When a client connects, the daemon forks a new address space to handle the client session. In that new address space an execv() is done to load the ftpdns module, and the configuration parameters are passed in a parameter list over the execv(). Refer to Figure E-1 for an overview.

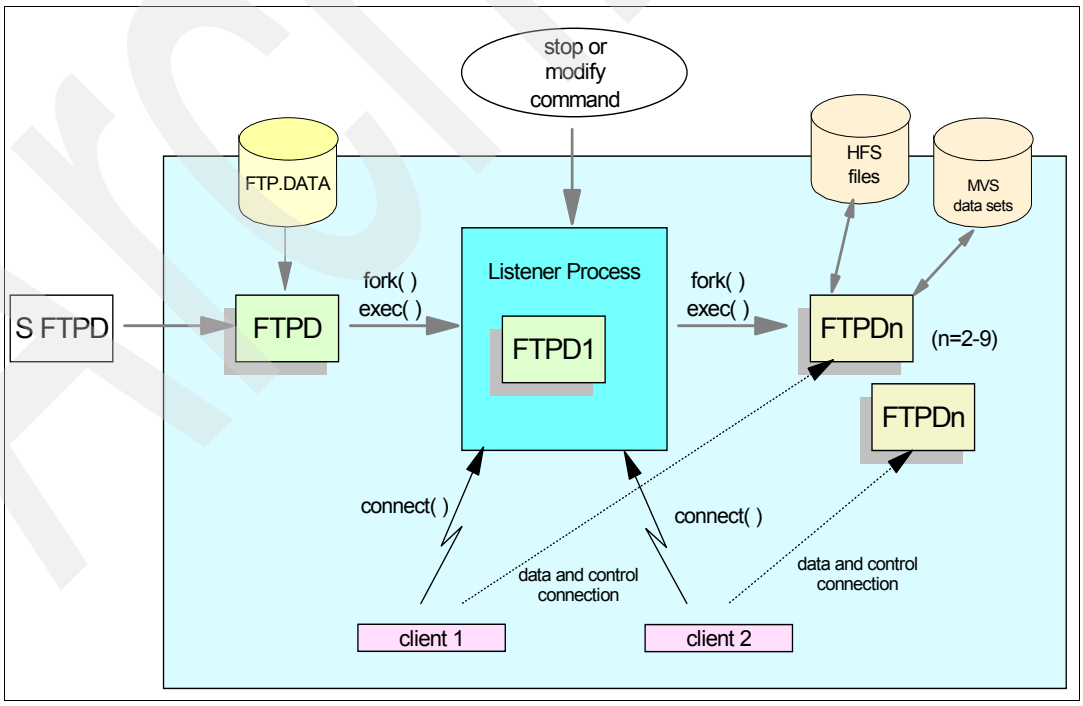


Figure E-1 Process flow of the server

When a new FTP client connects to the FTPD daemon process, the FTP daemon forks another address space that uses the execv() services to start the connection-specific server program, the ftpdns program. When the FTP daemon process forks an FTP server process, a new job name is generated by UNIX System Services. If the original job name is less than 8 characters, UNIX System Services adds a digit between 1 and 9. At TCP/IP startup this will always be a 1. So in most cases the FTP daemon address space would be named FTPD1. If your original job name is 8 characters, UNIX System Services uses the same job name for the background job. Similarly, every instance of the FTP server address space will have a name generated by UNIX System Services. Refer to *z/OS UNIX System Services Planning*, GA22-7800 for details about how job names are generated in UNIX System Services.

Because each client gets associated with its own copy of the FTP server, each client can set its own translation parameters and not adversely affect the other client/server sessions in effect.

SITE and LOCSITE commands and their parameters

Each operating system has unique requirements for allocating files or data sets in its file system. These requirements differ so widely between operating systems that it has been impossible to develop a single protocol that embraces all requirements from all operating systems.

In order to cover all requirements, the FTP protocol implements a **`SITE`** command, which enables an FTP client to send imbedded parameters to the FTP server over the control connection.

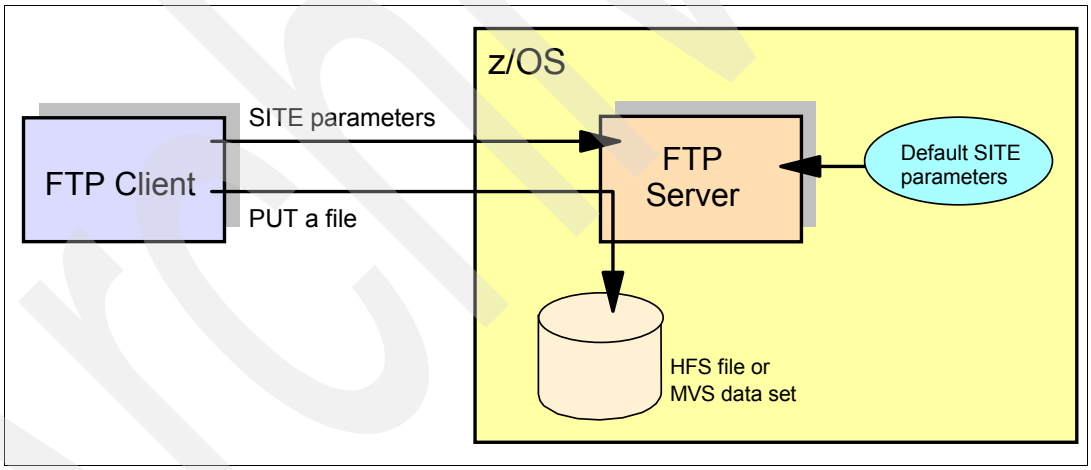


Figure E-2 When to use SITE parameters - z/OS FTP server

When an FTP client issues a `put` to transfer a file to the z/OS FTP server, the FTP server needs specific parameters in order to allocate a data set. These parameters include record format (RECFM), record length (LRECL), unit-type (UNIT), and blocksize (BLKSIZE), plus many others, depending on the specific operation requested. The FTP server has a set of default values for all the parameters it may need. The client can change many of these values for the current FTP session via the **`SITE`** command.

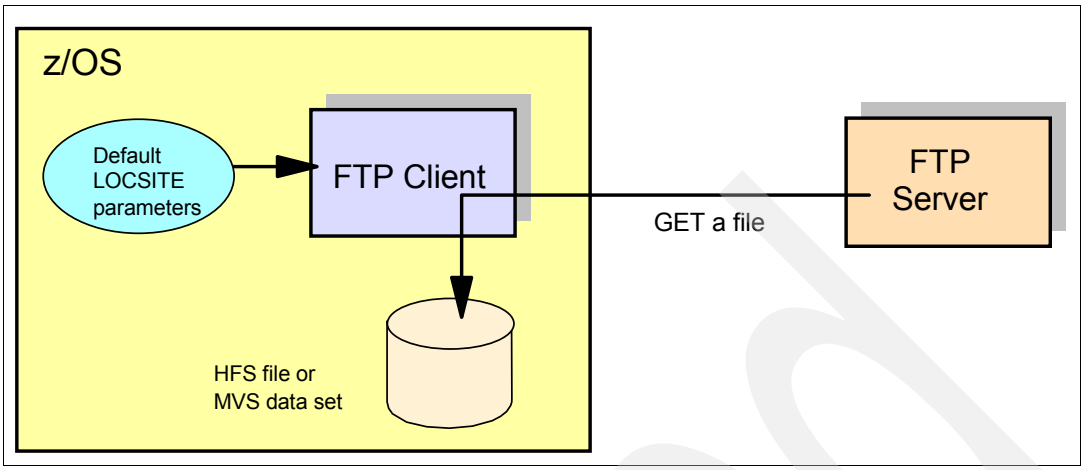


Figure E-3 When to use LOCSITE parameters - z/OS FTP client

If you use the z/OS FTP client function and you retrieve a file from an FTP server somewhere in your IP network, the FTP client function also needs a set of parameters similar to those of the z/OS FTP server, in order to allocate a data set in MVS. Again a set of default values exists for the z/OS FTP client, but you as a user may change these via a **LOCSITE** command.

Specification of FTP default values

Default values for data set and disk parameters can be specified using the FTP configuration data sets.

For the FTP client, the default LOCSITE parameters are specified in an installation-wide default configuration data set, or in a user-specific configuration data set. For the FTP server, the default SITE parameters are specified in an FTP server configuration data set.

Default FTP Client LOCSITE parameters

When a user on your z/OS system starts the FTP client function, a set of default local SITE parameters are in effect. The user can change these parameters during the FTP session by using the **LOCSITE** command.

The FTP client function searches for a configuration data set that contains the desired default parameters. The search order is defined in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

If none of these data sets is found, the FTP client will use a set of hard-coded default values for the local SITE parameters.

If a user needs to use a different setup, that user may make a temporary modification by means of the **LOCSITE** command, or a permanent modification by creating a userid.FTP.DATA configuration data set. Review the hlq.SEZAINST(FTCDATA) sample member for client SITE parameters.

Note: Not all SITE parameters can be specified in the FTP.DATA file and not all parameters specified in the FTP.DATA file can be changed with SITE or LOCSITE commands.

For details about both the SITE, LOCSITE, and FTP.DATA client statement parameters, consult *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780.

Default FTP Server SITE parameters (FTPDATA)

The FTP server uses a set of default SITE parameters.

When a client connects to the z/OS FTP server, the default SITE parameters will become the actual SITE parameters for that FTP session, unless the client during the session changes them by means of an FTP client **SITE** command.

> **Note:** Some systems do not support the **SITE** command. In order to pass the SITE parameters to MVS, the user has to enter the **QUOTE** command in front of SITE. To pass, for example, the RECFM=U parameter to MVS, the user might have to type in:
>
> ```
> quote site recfm=u
> ```

During startup, the server will look for your default SITE parameters in an FTP configuration data set. The search order for FTP.DATA for the server is documented in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

The search step for FTP.DATA that uses a job name as high-level qualifier will use the original job name and not the UNIX System Services generated FTP daemon job name. This original job name will be used to search for both FTP.DATA and translation tables, for example FTPD.FTP.DATA and FTPD.STANDARD.TCPXLBIN. If the ftpd daemon program were started from the UNIX System Services shell, these search steps would use the user ID of the process that started the listener program or the value of the _BPX_JOBNAME environment variable. The *datasetprefix* used in the last search step comes from the UNIX System Services resolver configuration file, in our setup the SYSTCPD DD statement. If none of these data sets is found, the server will use a set of hard-coded default values. Review the hlq.SEZAINST(FTPSDATA) sample member for server site parameters. Also review the *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for details about the FTP.DATA configuration statements.

How can FTP translation be applied?

TCP/IP uses the following types of translation tables:

- Single byte character set (SBCS) translation tables are used for sing-byte characters. This includes most languages.
- Double-byte character set (DBCS) translation tables are used for converting double-byte characters. DBCS translation tables are required for character sets such as Japanese Kanji, and some Chinese sets, which contain too many characters to represent them by using single-byte codes. SBCS translation tables provide mappings for a maximum of 256 characters. DBCS translation tables can provide up to a theoretical maximum of 65,535 character mappings; however, DBCS character sets usually contain less than this.
- Multi-byte character set (MBCS) translation tables are used in supporting the Chinese standard GB18030. This support is provided by iconv with code page IBM-5488, and does not allow for customized tables. See the ENCODING and MBDATACONN statements in the FTP.DATA file that define this support for FTP in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. Also refer to *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780, to see how to use the LOCSITE and SITE client commands to specify ENCODING and MBDATACONN during an FTP session.

Supported translations

The various languages and encodings supported by FTP (both SBCS and DBCS) are documented in tables in the following product manuals:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

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

In a z/OS FTP client session, the user can select which DBCS translation to use via the DBCS subcommands of the FTP client function. Similarly, the user can issue FTP subcommands to select Unicode data transfer and conversion to/from EBCDIC.

The z/OS FTP server accepts the **TYPE** subcommand with parameters that can specify DBCS or Unicode translation for the data transfer.

For SBCS translation, the z/OS FTP client user selects a translate table via the FTP command line keyword TRANSLATE, and the z/OS FTP server accepts a **SITE** command with a translate table name from a remote FTP client.

Translation table search orders

The search order that FTP uses to find a specified translation table are documented in *z/OS Communications Server: IP Configuration Reference*, SC31-8776. A separate search order exists for each of the two data streams: one for SBCS and a another for DBCS.

A user selects DBCS tables independently of SBCS tables. If a remote FTP client wants to transfer a mixed-mode data stream (a data stream that consists of both DBCS and SBCS data sections intermixed with shift-in and shift-out control characters), the user has to select a DBCS translation table using an **FTP TYPE** command. The user may change an SBCS translation table using a **SITE SBDATACONN** command. The selected DBCS table will be used to translate the DBCS sections, and the SBCS table will be used to translate the SBCS sections of the mixed-mode data stream.

Both the FTP server and the FTP client allow the use of separate SBCS translation tables for the data connection and the control connection.

If you use the FTP client and want to change the code pages of the client, you have to use the **LOCSITE FTP** subcommand instead of the **SITE** subcommand.

Transfer mode and data type

It may seem to be a trivial matter to transfer files between different computer systems, but when you take a closer look, you soon discover a range of issues originating from the diversity of computer architectures represented in a typical IP network. Some operating systems use 7-bit ASCII to represent character data, others use 8-bit ASCII or EBCDIC, just to mention the most obvious. Some operating systems organize files into records, while others treat files as continuous streams of data, in some situations without any encoded notion of record boundaries (in such a case, it will be up to the program reading or writing the data to impose a structure onto the data stream).

The FTP protocol deals to a great extent with these issues, but you, as a user of this protocol, have to select the proper options in order to let FTP transfer a file in such a way that it is usable on the receiving system.

FTP always transfers data in 8-bit bytes; this is called the *transfer size*. If the sending or receiving system uses another byte length, it is up to the FTP client and the FTP server to implement the proper conversion between local byte sizes and the FTP transfer size. When FTP transfers ASCII data, it always transfers it in 8-bit bytes, where the bits are used to encode the ASCII character according to a specific ASCII standard, which is called NVT-ASCII (Network Virtual Terminal ASCII as defined in the TELNET protocol). This implies that when you transfer ASCII type data between two ASCII hosts, a translation from the local ASCII representation to NVT-ASCII for transmission and back to the receiving hosts local ASCII representation always takes place.

When MVS is involved in an ASCII type transfer, MVS will translate the received NVT-ASCII into EBCDIC, and translate data to be transmitted from EBCDIC into NVT-ASCII.

When you request an FTP file transfer, you can characterize the transfer by means of three attributes:

TYPE The term used in the official sources is *data type*, but you may also see the term *transfer type* or *representation type*. Data type is used to signal how the bits of the transmitted data should be interpreted by the receiver.

Data type will normally have one of the following three values:

ASCII When you set the data type to ASCII, the receiver knows that the data is character data and that each line of data is terminated via a control sequence of Carriage Control plus Line Feed (CRLF), which in ASCII is X'0D0A'. If MVS is the receiving side, data will be translated from NVT-ASCII to EBCDIC and the CRLF sequences will be removed from the data stream and substituted by traditional MVS record boundaries according to the current settings of the SITE/LOCSITE parameters: RECFM and LRECL. If RECFM is fixed, the data records will be padded with extra spaces in order to fill up a record. If MVS is the sending side, the data will be translated from EBCDIC into NVT-ASCII and, based on existing record boundaries, CRLF sequences will be constructed and inserted into the ASCII data stream.

A data type of ASCII is the default data type in all FTP implementations.

EBCDIC A data type of EBCDIC means that the data transferred is EBCDIC data. In such a case, no translation to NVT-ASCII or from NVT-ASCII takes place in MVS. The 8-bit EBCDIC bytes are transferred as they are. If you transfer text data between two EBCDIC systems, a data type of EBCDIC is the most efficient way to transfer the data. Most ASCII hosts will reject a data transfer where you specify a data type of EBCDIC. Some will treat it as an ASCII transfer, but the point where the translation takes place is at their end of the FTP connection, and not in MVS.

IMAGE A data type of IMAGE means that the data will be transmitted as contiguous bits packed into the 8-bit FTP transfer byte size. No translation takes place, neither at the sending nor at the receiving side. You will normally use this data type for binary data, such as program files. If you transfer text data between two similar ASCII hosts, it will often be more efficient to use an IMAGE data type instead of an ASCII data type. As the two systems use exactly the same ASCII representation, there is no need to impose the overhead of translating to and from NVT-ASCII.

Note: Other data types are all flavors of DBCS and Unicode.

Using the RFC2389 and RFC2640 FTP features

These RFCs represent the feature negotiation mechanism and FTP internationalization respectively.

RFC 2389 - Feature negotiation

The Feature negotiation mechanism enables FTP clients to ask the server which features or options it supports. Commands used with this feature are:

- On the z/OS Server

```
   OPTS
FEAT
```

- On the z/OS Client

```
feature
```

- On the non z/OS Client

```
quote feature
```

Displaying server features depends upon what is included in the EXTENSIONS statement in FTP.DATA. Figure E-4 shows an example of a FTP.DATA showing some extension statements.

```
EXTENSIONS      MDTM       ; Server can respond to MDTM cmd
EXTENSIONS      UTF8       ; Server can respond to LANG & UTF-8 enc.
EXTENSIONS      REST_STREAM  ; Server can respond to SIZE cmd
```

Figure E-4 FTP.DATA file showing some of the extensions supported

RFC2640 - FTP Internationalization

FTP servers use English alpha numerics for their directory and file names, even though the site serves a non-English-speaking community. This is so because FTP constrains pathnames to single byte encodings (ASCII). With Internationalization, 7-bit restrictions on pathnames used in client commands and server responses are removed. UCS transformation format UTF-8 is used, and a new command for language negotiation is defined.

To activate the RFC2640 implementation, the EXTENSIONS UTF8' parameter in TCP is required in FTP.DATA, as shown in Figure E-4 on page 378. This parameter has to be coded in both the z/OS Client and the z/OS Server. A client can thus override statements within FTP.DATA which control code page selection by using the LOCSITE CONTRLCONN subcommand.

Requirements to implement these RFCs

RFC 2389 is available in z/OS Communications Server with no additional tasks having to be performed to activate the support.

RFC 2640 requires you to code an EXTENSIONS UTF8 statement in your server and clients FTP.DATA

The National Language Resources component of z/OS Language Environment must be installed for the UTF-8 function to be available.

Once UTF-8 has been negotiated between client and server, the STAT command will indicate that UTF-8 is active on the control connection. Figure E-5 on page 379 shows the results of a successful UTF-8 negotiation.

```
stat
 >>> STAT
 211-Server FTP talking to host 9.12.6.30, port 1088
 211-User: GARTHM  Working directory: GARTHM.
 211-UTF-8 encoding in use on the control connection
 211-The control connection has transferred 309 bytes
 211-There is no current data connection.
 211-The next data connection will be actively opened
 211-to host 9.12.6.30, port 1088,
 211-using Mode Stream, Structure File, type ASCII, byte-size 8
 211-Automatic recall of migrated data sets.
 211-Automatic mount of direct access volumes.
 211-Auto tape mount is allowed.
 211-Inactivity timer is set to 300
 211-VCOUNT is 59
 211-ASA control characters in ASA files opened for text processing
 211-will be transferred as ASA control characters.
 211-Trailing blanks are removed from a fixed format
 211-data set when it is retrieved.
```

Figure E-5 STAT command output

Selecting translation tables

If you want to transfer files that contain DBCS data, you have to configure the TCPIP.DATA file with a LOADDBCSTABLES statement with the proper DBCS code names. The following are examples of how to select the translation tables. In all of the examples, we have configured the following statement in the TCPIP.DATA used by the FTP server and client.

```
DATASETPREFIX TCPIP
LOADDBCSTABLES SJISKANJI EUCKANJI
```

Using the QUOTE SITE subcommand

The FTP server has been started by the cataloged procedure named FTPDB and the FTP daemon address space had a UNIX System Services generated name FTPDB1.

```
ftp> quote site ctrlconn=7bit   1
200 Site command was accepted
ftp> quote site sbdataconn=garthm.ftpkana.tcpxlbin  2
200 Site command was accepted
ftp> quote type b 1             3
200-Representation type is KANJI Shift-JIS
200 Standard DBCS control used
```

Figure E-6 Selecting translation tables

1 The FTP client changes the FTP server translation table for the control connection to 7-bit ASCII. Since some systems may not recognize the **site ctrlconn** subcommand, you would have to use the **quote** FTP subcommand.

2 The FTP client tries to change the translation table for the data connection used by the FTP server to a customized SBCS table.

3 The FTP client asks the FTP server to change the current transfer type to Shift JIS Kanji. The message with the number 200 will be returned to the client, if the FTP server server loads the corresponding DBCS translation table correctly.

If you specify an invalid parameter or the FTP server cannot support the specified translation table, you will see the error message shown in Figure E-7.

```
ftp> quote site ctrlconn=8bit
200-ctrlconn=8bit ignored.  Requested conversion is not supported.
200 Site command was accepted
```

Figure E-7 site ctrlconn command on client

If you specify an invalid table name, the FTP server will return the error message shown in Figure E-8 to the client.

```
ftp> quote site sbdataconn=tcpip.notable
200-Translate file 'tcpip.notable' not found.  SBDATACONN ignored.
200 Site command was accepted
```

Figure E-8 site sbdataconn command on client

If the LOADDBCSTABLES statement is not specified in the TCPIP.DATA file that is allocated for the FTP server, or the proper code pages are not specified to the LOADDBCSTABLES statement, the attempt will fail and the client will receive the message shown in Figure E-9.

```
504-Type not Supported. Translation table not Loaded.
504 Type remains Ascii NonPrint
```

Figure E-9 Type not supported error

Using the TRACE option at the server

If you set the TRACE option for the z/OS FTP server, the trace entries shown in Figure E-10 on page 381 will be written to the syslog file (if daemon.debug has been defined in /etc/syslog.conf) or displayed on the MVS console if syslogd is not active:

Archived

```
get_command: select rc is 1
get_command: received 20 bytes
Parse_cmd: command line: site ctrlconn=7bit.  1
site_cmd entered with 'ctrlconn=7bit'
site(): processing ctrlconn=7bit
site: freeing site arg: 'CTRLCONN'
get_command: select rc is 1
get_command: received 40 bytes
Parse_cmd: command line: site sbdataconn=kakky.ftpkana.tcpxlbin.  2
site_cmd entered with 'sbdataconn=kakky.ftpkana.tcpxlbin'
site(): processing sbdataconn=kakky.ftpkana.tcpxlbin
site: freeing site arg: 'SBDATACONN'
get_command: select rc is 1
get_command: received 10 bytes
Parse_cmd: command line: type b 1.
xtype: xtype routine entered with type 'B' dataformat '1' opt1 ' ' opt2
trying to open DBCS file //'FTPDB.SRVRFTP.TCPKJBIN'       3
fopen failed for //'FTPDB.SRVRFTP.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
trying to open DBCS file //'TCP.SRVRFTP.TCPKJBIN'         3
fopen failed for //'TCP.SRVRFTP.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
trying to open DBCS file //'FTPDB.STANDARD.TCPKJBIN'      3
fopen failed for //'FTPDB.STANDARD.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
NC1444 trying to open DBCS file //'TCP.STANDARD.TCPKJBIN' 3
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
EZY2720I Using Japanese translation tables in 'TCP.STANDARD.TCPKJBIN' 4
```

Figure E-10 syslogd output

1 The FTP client changes the FTP server translation table for the control connection to 7-bit ASCII.

2 The FTP client changes the FTP server translation table for the single byte data connection to a private translate table.

3 You can see the search order for the DBCS translation table. In this case, the name of the FTPD cataloged procedure is FTPDB, so FTPDB will be used as the original job name.

4 The DBCS translation table loaded by the FTP server.

Using the DEBUG option at the client

Figure E-11 shows examples of messages displayed when we started FTP clients with the debug option. You can use either -d or (TRACE for this option.

```
EZA1736I FTP -d (EXIT
EZYFT18I Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages. 1
EZY2640I Using dd:SYSFTPD for local site configuration parameters.
EZA2794I Both CCTRANS and CTRLCONN were specified. Using CTRLCONN.        2
                                        CCTRANS will be ignored.
EZA2795I Both SBTRANS and SBDATACONN were specified. Using SBDATACONN.    3
                                        SBTRANS will be ignored.
EZYFT27I Using conversion between 'IBM-850' and 'IBM-1047'
                                        for the control connection.      4
EZYFT31I Using TCP.FTPKANA.TCPXLBIN for FTP translation tables
                                        for the data connection.         5
EP4873 set_dbcs: __ipdbcs() returned 2 parms from LOADDBCSTABLES statement(s)
EP4879 main: dbcstables->__ip_dbcs_listÿ0Ã is: SJISKANJ.  Len is: 8       6
EP4879 main: dbcstables->__ip_dbcs_listÿ1Ã is: EUCKANJI.  Len is: 8       6
EZA1450I IBM FTP CS/390 296 00:10 UTC
EZA1466I FTP: using TCPIPA instead of INET
EZA1772I FTP: EXIT has been set.
EZA1456I Connect to ?
EZA1736I 9.24.104.26
```

Figure E-11 FTP client with debug option

1 The messages used by the FTP clients are stored in message catalogs to allow message translation.

2 If both the CCTRANS and CTRLCONN keywords are specified in FTP.DATA, CTRLCONN is preferred.

3 If both the SBTRANS and SBDATACONN keywords are specified in FTP.DATA, SBDATACONN is preferred.

4 The translation used by the client for control connection.

5 The translation table used by the client for data connection.

6 The DBCS code pages specified in the TCPIP.DATA configuration data set have been allocated for this client. Each code page name is recognized by the preceding 8 characters.

Using the TRANSLATE sub command

If you specify the **TRANSLATE FTP** command option, the *hlq*.STANDARD.TCPXLBIN data set is never used. If the translation table search procedure using a data set name specified with this option fails, then FTP ends with an error message as shown in Figure E-12 on page 383.

```
KAKKY@/u/kakky$ftp -d 9.24.104.47 \(translate notable  1
Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
Using FTP configuration defaults.
NX0570 read_xlate_files: Unable to open /u/kakky/notable.tcpxlbin :  2
                  EDC5129I No such file or directory.
NX0613 read_xlate_files: Unable to open //'KAKKY.NOTABLE.TCPXLBIN' :
                  EDC5049I The specified file name could not be located.
NX0625 read_xlate_files: Unable to open //'TCP.NOTABLE.TCPXLBIN' :
                  EDC5049I The specified file name could not be located.
Cannot load translate table specified by TRANSLATE parameter notable 3
```

Figure E-12 Translate table not found

1 If you issue the FTP command in the UNIX System Services shell with MVS style options, you have to precede the left parenthesis with an escape character such as the backslash(\).

2 In the UNIX System Services environment, the translation table in the Hierarchical File System is the top of the table search hierarchy.

3 The FTP command stops. No more table search attempts are executed.

Setting a DBCS transfer mode

When you use the DBCS data transfer mode in the FTP client, the debug option of the FTP client will show you the messages displayed in Figure E-13 on page 384.

Archived

```
EZA1460I Command:
sjiskanji (notype  1
 PC0291 Input to parCmd is :
 PC0293 parseCmd: Number of parameters is 2
 PC0296 parseCmd: parameter 0 is sjiskanji
 PC0296 parseCmd: parameter 1 is (notype
 PC0486 fndCmd: entering with sjiskanji.
 PC0568 fndCmd: Command found is sjiskanji
 PC0305 parseCmd: fndCmd returned the cmdrecord for sjiskanji
 PC0386 parseCmd: Using primary session
 CT2205 sjiskj: routine entered with parmcount=2
 NC1017 get_client_dbcs_table() entered for lang type 1
 NC1053 trying to open DBCS file //'KAKKY.FTP.TCPKJBIN'          2
 NC1079 fopen failed for //'KAKKY.FTP.TCPKJBIN'.
                EDC5049I The specified name could not be located.
 NC1053 trying to open DBCS file //'TCP.FTP.TCPKJBIN'            2
 NC1079 fopen failed for //'TCP.FTP.TCPKJBIN'.
                EDC5049I The specified file could not be located.
 NC1053 trying to open DBCS file //'KAKKY.STANDARD.TCPKJBIN'     2
 NC1079 fopen failed for //'KAKKY.STANDARD.TCPKJBIN'.
                EDC5049I The specified file name could not be located.
 NC1053 trying to open DBCS file //'TCP.STANDARD.TCPKJBIN'       2
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1061 Using Japanese translation tables in 'TCP.STANDARD.TCPKJBIN' 3
 CT0334 entering cliDBOptopns() for newftpoptformat=0, parmcount=2
 CT0711 cliDBOpt: no cmd sent to server
 PC0452 parseCmd: Using primary session.
 EZA1460I Command:
```

Figure E-13 Debug log when using DBCS transfer mode

1 Change the data transfer mode to Japanese Shift JIS Kanji. In order to suppress the sending of the corresponding TYPE message to the FTP server which may not recognize it, the end user issues the SJISKANJI FTP subcommand with the NOTYPE parameter.

2 The debug message shows you the DBCS translation search order used by the FTP client.

3 The DBCS translation table loaded by the FTP client.

Enabling Unicode transfer mode

Figure E-14 illustrates how to enable Unicode data transfer mode. In this example, we used a PC FTP client and z/OS FTP server.

```
ftp> quote type u 2   1
200 Representation type is UCS-2
ftp> quote site noucstrunc noucssub ucshostcs=ibm-939  2
200 Site command was accepted
ftp> put jpdata.ucs 'kakky.test.jpdata'
200 Port request OK.
125 Storing data set KAKKY.TEST.JPDATA
250 Transfer completed successfully.
local: jpdata.ucs remote: 'kakky.test.jpdata'
96 bytes sent in 0.00 seconds (0 Kbytes/s)
```

Figure E-14 Unicode data transfer mode

1 Issue **TYPE U 2**. Receiving this **TYPE** command, the FTP server sets the transfer type to UCS-2 TYPE. Since the PC client cannot recognize this TYPE, you have to precede the FTP subcommand with **QUOTE**.

2 Specify the EBCDIC code set to be used for converting to/from Unicode. In this sample, we use the IBM-939 code character set. You have to specify the EBCDIC code set using the code character set name supported by the iconv() API. At the same time, we disable the substitution and truncation. If you transfer the EBCDIC data that contains DBCS characters, you should disable the truncation, or the EBCDIC data might collapse.

Verify the server status using **STAT FTP** subcommand. The output shown in Figure E-15 is part of the output from the **STAT** subcommand.

```
211-using Mode Stream, Structure File, type UCS-2, byte-size 8
211-TYPE U data will be converted to/from IBM-939       1
211-UCS Substitution: OFF, UCS Truncation: OFF          2
211-Byte Order: big-endian                              3
```

Figure E-15 Partial STAT command output

1 The data encoded in Unicode will be converted to the IBM-939 character set.

2 Neither the UCS substitution nor UCS truncation are enabled. If you transfer SBCS/DBCS mixed data, then disable the UCS truncation.

3 The byte order for the Unicode encoding is set to big-endian by default. You can change it to the little-endian byte order by issuing the **TYPE U 2 L** FTP subcommand.

If you want to use the z/OS FTP client in Unicode transfer type, you have to issue the **UCS2** FTP subcommand and optionally the **LOCSITE** subcommand with UCS* parameters before starting the file transfer. You can also verify the status of the FTP client with the **LOCSTAT FTP** subcommand.

Archived

Archived

F

Our implementation environment

We wrote the four *Communications Server for z/OS TCP/IP Implementation* publications at the same time. Given the complexity of this project, we needed to be somewhat creative in organizing the test environment so that each team could work with minimal coordination and interference from the other teams. In this appendix we show the complete environment used for the four books, and the environment used for this IBM Redbooks publication.

Archived

© Copyright IBM Corp. 2008

The environment used for all four books

To enable concurrent work on each of the four books, we set up and shared the test environment illustrated in Figure F-1.

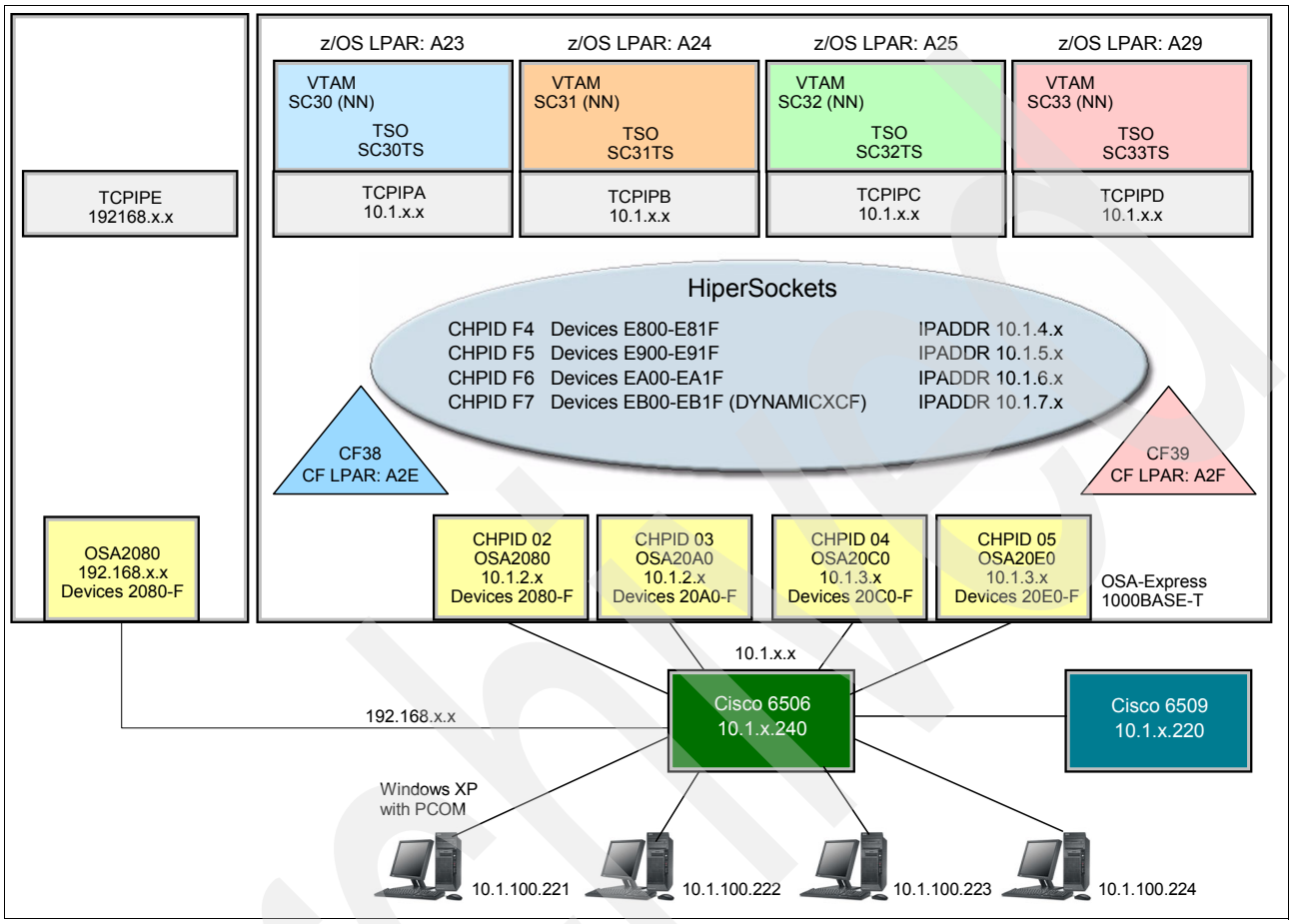


Figure F-1 Our implementation environment

Our books were written (and our implementation scenarios were run) using four logical partitions (LPARs) on an IBM System z9™ EC. These LPARs are referred to as LPAR A23, A24, A25, and A29.

We implemented and started one TCP/IP stack on each LPAR. Each LPAR shared the following resources:

- HiperSockets inter-server connectivity
- Coupling Facility connectivity (CF38 and CF39) for Parallel Sysplex scenarios
- Four OSA-Express2 1000BASE-T Ethernet ports cross-connected to a pair of Cisco 6500 switches

Note: This environment was based on the premise of high availability (no single points of failure).

Finally, we shared four Windows workstations, representing corporate network access to the z/OS networking environment. The workstations could be connected to either one of the Cisco switches. For verifying our scenarios, we used applications such as TN3270 and FTP.

The IP addressing scheme we used allowed us to build multiple subnetworks in order to avoid impeding any ongoing activities from other team members.

We also defined VLANs in order to isolate the TCP/IP stacks and portions of the LAN environment; see Figure F-2.

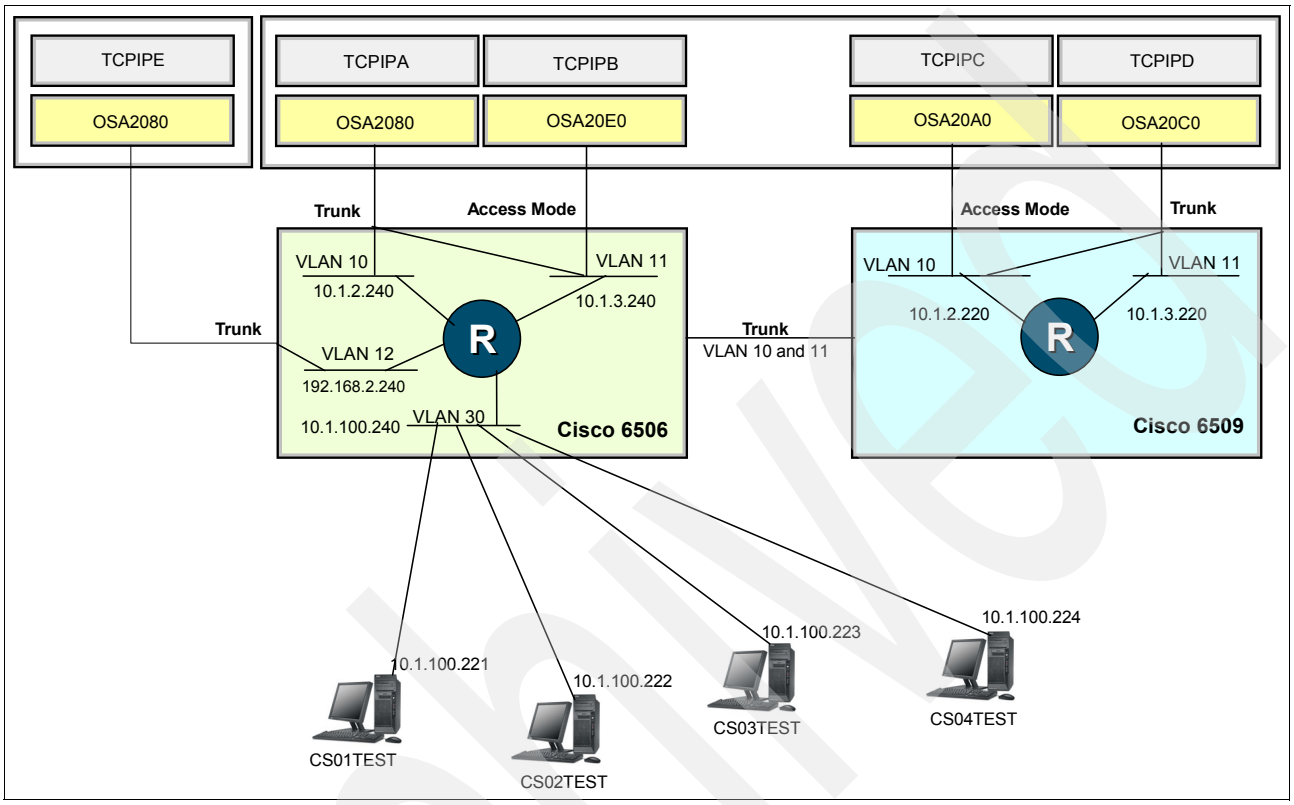


Figure F-2 LAN configuration - VLAN and IP addressing

Our focus for this book

Figure F-3 on page 390 depicts the environment that we worked with, as required for our standard application implementation scenarios.

Archived

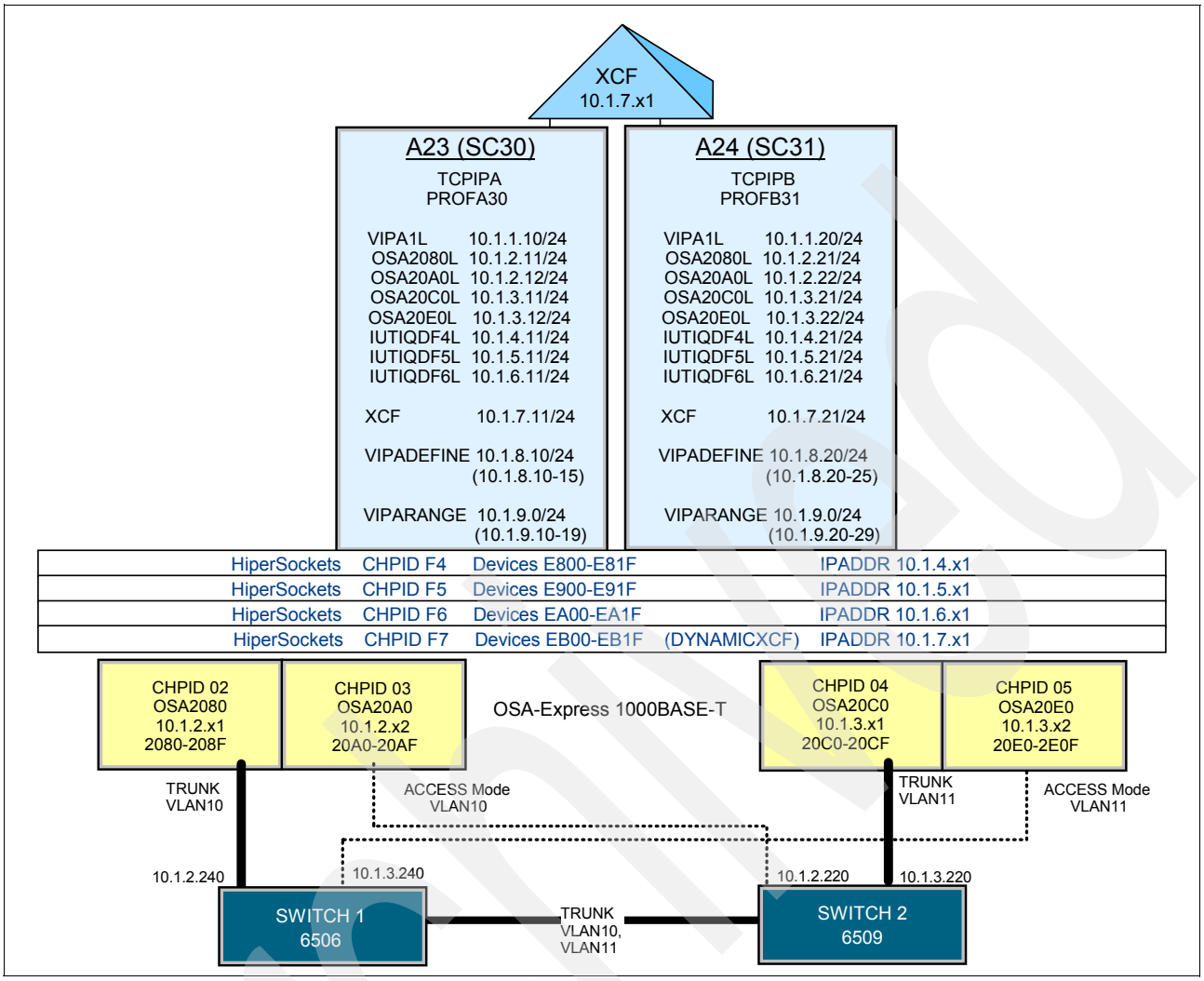


Figure F-3 Our environment for this book

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 393. Note that some of the documents referenced here may be available in softcopy only.

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 2: Standard Applications*, SG24-7533
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *IP Network Design Guide*, SG24-2580
- *OSA-Express Implementation Guide*, SG24-5948
- *zSeries HiperSockets*, SG24-6816
- *SNA in a Parallel Sysplex Environment*, SG24-2113
- *High Availability Considerations: SAP R/3 on DB2 for OS*, SG24-2003
- *z/OS 1.6 Security Services Update*, SG24-6448

Other publications

These publications are also relevant as further information sources:

- *z/OS XL C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS MVS IPCS Commands*, SA22-7594
- *z/OS MVS System Commands*, SA22-7627
- *z/OS TSO/E Command Reference*, SA22-7782
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS Communications Server: CSM Guide*, SC31-8808
- *z/OS Communications Server: New Function Summary*, GC31-8771
- *z/OS Communications Server: Quick Reference*, SX75-0124
- *z/OS Communications Server: IP and SNA Codes*, SC31-8791

Archived

© Copyright IBM Corp. 2008. All rights reserved.

- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference*, SC31-8788
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IPv6 Network and Application Design Guide*, SC31-8885
- *z/OS Migration*, GA22-7499
- *z/OS MVS System Commands*, SA22-7627
- *OSA-Express Customer's Guide and Reference*, SA22-7935
- *z/OS Communications Server: SNA Operation*, SC31-8779
- *z/OS TSO/E Command Reference*, SA22-7782
- *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services File System Interface Reference*, SA22-7808
- *z/OS UNIX System Services Messages and Codes*, SA22-7807
- *z/OS UNIX System Services Parallel Environment: Operation and Use*, SA22-7810
- *z/OS UNIX System Services Programming Tools*, SA22-7805
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS UNIX System Services User's Guide*, SA22-7801
- *sendmail 3rd Edition*, O'Reilly & Associates, Inc., ISBN 1-56592-839-3
- *termcap & terminfo*, O'Reilly & Associates, Inc., ISBN 0-93717-522-6

Archived

Online resources

These Web sites and URLs are also relevant as further information sources:

- z/OS Communications Server product support

  http://www-306.ibm.com/software/network/commserver/zos/support/

- Mainframe networking

  http://www.ibm.com/servers/eserver/zseries/networking/

- z/OS Communications Server product overview

  http://www.ibm.com/software/network/commserver/zos/

- z/OS Communications Server publications

  http://www-03.ibm.com/systems/z/os/zos/bkserv/r9pdf/#commserv

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Archived

Index

Symbols

/etc/inetd configuration file 187
/etc/mail directory 217
/etc/syslog.conf configuration file 6, 9

Numerics

7-bit ASCII 87, 197
  data 199
  image file 229
  text 199

A

abend trap 74
address space 19, 130, 163, 313
ADNR 294
AFP format 170
Agent and Subagent initialization messages 145
ahpseu 240
ahtcap 240
alias file 214, 219
ALLOWAPPL
  NVAS 153, 334, 346
  SC30N 153, 334, 346
application programming interface (API) 83
Area Area_Number 339, 354
ASCII 87, 372
ASCII character 17, 87, 240
ASCII data
  stream 88
  type 88
ASCII/EBCDIC translation 80
  translation facilities 80
assigned by z/OS UNIX 81
authoritative DNS server 283
  considerations 284
  dependencies 283
  description 283
AUTOLOG
  statement 100, 252, 257
    RXSERVE name 252
AUTOLOG and PORT statements 100, 252
automated domain name registration 294

B

backup stack to support Sysplex functions 58
Banner Pages 239
BASEWLM algorithm 57, 111, 352
basic concept 1–2, 79–80, 127, 159, 185–186, 197, 235, 279
basic port 334, 348
batch job 83, 119, 163, 200, 253, 263
  email 213, 227
  environment 267
  log 248
  REXEC TSO client command 265
BEGINVTAM block 20
Berkeley Internet Name Domain (BIND) 279
BIND 9 configuration file 286
BIND 9 DNS
  related file 288
BIND 9.2
  name server 283
  rndc 283
BPX_JOBNAME 81
BPX_JOBNAME variable 100
browse the log to see the latest query activity just created 294

C

Caching-only DNS server 282, 284, 294
  Considerations 285
  Dependencies 285
  Description 285
  Problem determination 304
Cataloged procedure for the FTP server 99
cftpcli 86
chargen service 194
  Verification 194
Check Client Connection status 46
Checking maximum connections supported 34
Checking the total number of TN3270 ports defined in the profile 35
chmod 777 219
client 249
client auth 346
client authentication 20, 346
client certificate 20, 347
client ID 25
client LPAR 97
CLIENTAUTH SSLCERT 347
clientid group 26
  only member 27
clientid specification 26
clientid summary 37
command issues 28
command line 6, 202, 241, 249
  interface 207
  MUA 214
  option 241, 243
  parameter 100, 187, 232, 242
  security information 249
  switch 216
command parameters for VARY TCPIP 32
complete detail xi
complete name 281
compose an RFC 822 compliant message 228

© Copyright IBM Corp. 2008. All rights reserved.

concurrent execution for TSO and z/OS UNIX remote execution servers 260
configuration example 15, 235
configuration file 2–3, 25, 81, 133, 172–173, 187, 189, 214, 238, 283, 285, 315, 324, 331, 343
    last four lines 6
configuration statement 53
configure logging options for the server 289
configure syslogd 101
configure the Query Engine started task (optional) 138
configure the Trap Forwarder started task (optional) 139
CONN (CS) 35
CONN Identifier 42
connection and session management 20
connection diagnostics 21
connection mapping 20, 26
connection mapping methodology 26
Connection Security 18
Connection security 18
CONNTYPE Basic 346
considerations 84, 109, 123, 213
copy the sample configuration file 217
Coupling Facility (CF) 54, 108
CRLF sequence 88
    ASCII data type 88
crontab entry for syslogd 7, 10
CS06.TCPP Arm 29, 264
CTRACE option 76, 319
    complete discussion 76
CTRACE system component to trace TN3270 processes 76
CTRLCONN 376
CURR CONNS 36
current dynamic VIPA configuration 66, 116

D

data connection 81, 316
data set 8, 24, 84–85, 100, 130, 136, 168, 191, 202, 250, 268, 287, 321
data stream
    USS table 324
data structure 88, 376
data trace to capture socket data 76
data type 87–88, 376
    supported combinations 89
data type, structure and mode 87
dataset attributes 85
daytime service
    Verification 193
DD statement 135, 171, 204
DD SYSOUT 7, 31, 135, 288, 332, 344
debug message 21, 317
    several types 21
debug option 70
    obeyfile command 73
debug statement 71
default router 59, 339, 354
DEFAULTLUS 334, 346
define a relay server if one is planned 211
define system security for the tn3270 started task 29
definition statements within the profile 70
DEFLT APPL 42
dependencies 84, 108, 123, 213
description 213
destination address 285
DESTIP 10.20.20.100 57, 111, 352
detailed information 62, 157, 277, 309
    dynamic VIPA 63
DG GENERALUSER 63
directory mode or data set mode 86
discard service
    Verification 193
display CLIENTID command to view client ID information 37
DISPLAY commands to view the status of TN3270 resources 35
display CONN 40
display CONN command to view connection information 40
display OBJECT command to view object information 39
display PROFILE to view profile information 35
DISTMETHOD ROUNDROBIN 57, 111
DISTMETHOD SERVERWLM 57, 111
distributed programming interface
    corresponding request 130
distributed programming interface (DPI) 131
DNS
    additional information sources 309
DNS name 159, 262
DNS service 211, 262, 284
Domain Name
    Server 231, 281, 336, 350
    Service 279
    System 199, 279
double-byte character
    set 82, 239
D-VIPA use 60, 340, 355
Dynamic VIPA 54, 109, 249
    configuration 66, 116
    configuration information 66
    configuration VIPADCFG 66, 116
    Connection 64, 114
    definition 54
    Destination Port Table 65, 115
    Destination Port Table information 65
    Destination Port Table VDPT 65, 114
    interface 59
    IP subnet 56
    Range requirement 60, 340, 355
    requirement 59
Dynamic VIPA configuration
    information 66, 116
    VIPADCFG 66, 116
Dynamic VIPA Connection 64, 114
Dynamic VIPA Destination Port Table 65–66, 114
DYNAMICXCF stmt 60, 341, 356

E

EBCDIC 372
EBCDIC system 88

text data 88
echo service
Verification 193
ECSALIMIT 0M 335, 349
e-mail address 222, 227
email message 226, 229
encode the attachment 228
ENCODING 93
ENCRYPT SSL_DES_SHA 347
ENDDESTIPGROUP 153, 347
end-of-file (EOF) 89
end-of-record (EOR) 88
environment variable 140, 170, 241, 250, 287, 311
file 287
etc/ftpbanner file 98
etc/inetd.conf file 220, 258, 325
etc/mail directory 213
Express Logon Feature (ELF) 20
EXPRESSLOGON 42
EZB0789I GetNextNote 177
EZB3170I Name 292
EZB3172I Address 292
EZZ0316I Profile 34, 179

F

FACILITY class 30, 98, 135, 189, 204
features of FTP 82
File Transfer Protocol (FTP)
ASCII 372, 377
CTRLCONN 376
data set mode 86
data structure 376
data type 376
directory mode 86
EBCDIC 372, 377
FTP.DATA 375
IMAGE 377
image transfer 377
LOCSITE parameter 373–374
QUOTE 375
record structure option 92
SBDATACONN 376
SITE 373
transfer mode 376
TRANSLATE 372, 376
translate tables 372
unicode data transfer 385
Web browser support 379
FLG 65, 115
foreign socket 150, 165, 192, 222, 291
FTP 80
a generic server in a multiple stack environment 123
FTP client 79–80
API 122
issue 85
support 91
FTP client via batch 119
FTP client via the FTP client API 122
FTP network management interface with SMF 123
FTP record structure option 92
FTP Server
Jobname 81
Start 101
FTP server 79, 81, 87, 336, 350
AUTOLOG statement 100
control connection port 81
data connection port 81
dependencies for Multiple servers with SD 109
PROFILE.TCPIP 100
sample FTP.DATA 81
FTP server configuration profile data set 113
FTP server started task procedure 113
FTP server with Sysplex Distributor
problem determination 124
FTP session 80
FTP transfer 87
FTP without security 84, 97
FTP.DATA 81, 85, 375
FTP.DATA dataset 85
FTP.DATA for the client 97
FTP.DATA for the server 98
fully qualified domain name (FQDN) 281

G

Generic Resource (GR) 21
Global_Options Ignore_Undefined_Interfaces 59, 339, 354
GWM 300

H

help file 215
hints file (root server) 288
hlq.SEZA INST 135, 204, 253
sample procedure 253
sample user exit 212
host IP 54, 109
host name 177, 212, 275, 281
Host on Demand (HOD) 15
http protocol 336, 350

I

implement the syslog daemon (syslogd) or alternative log file 290
implementation of the osnmp z/OS UNIX command 148
implementation scenario xi
implementation task 188, 217, 227, 238
INACT command 45
inactive LU 46
incoming server 223
individual parameter xi
INETD 195
A simple design 188
INETD configuration file changes for otelnetd 242
Infoprint Central 169
Infoprint Server 130, 160, 167
administering 172
advantages 169
complete implementation details 176

Archived

considerations 170
detailed implementation 170
IP printing 167
optional features 170
other advantages 169
Infoprint Server and other transforms 169
Infoprint Server components 170
Infoprint Server Windows Client 168
information technology 162
initiate or post (IP) 247
INT Table 42
interface display and traffic monitoring 147
interface statement 60, 340, 355
Internet Printing Protocol (IPP) 171
Internet Software Consortium (ISC) 279
IOBSNMP initialization and termination messages 146
IP address 54, 159, 187, 190, 242, 260, 262, 279, 340, 347
IP network 17, 86–87, 128, 159, 246
problem determination 157
SMTP servers 211
SNA 3270 traffic 17
IP printing 159
additional information sources 183
basic concepts 161
detailed information 183
importance 161
IP PrintWay 168–169
basic mode 171
FSAs 172
FSS definition 176
options section 174
printer definition 173
transmission-queue 172
IP programmer 130
IP routing 58, 111, 352
IP subnet 54, 109
IP subnet for Dynamic VIPA 56
IP subnet for XCF interfaces 56
ISP 209, 332, 344
ISPF panel (IP) 159, 198

J

JCL for the TN3270 started task 31
JES spool 161, 202, 257
data sets 169
output data sets 168
purge data sets 202
Select output data sets 171
job control language (JCL) 264
Job Entry Subsystem (JES) 160, 201, 264
jobname 4–5, 12, 81, 166, 187, 287, 313
msgs file 12
join the encoded file with the text message 229

K

Kerberos 242
Kerberos security
setting 316
table 317
key characteristic 1
KEYRING HFS 346

L

Language Environment
API 312
API variable 315
C Socket 313
dump 314
run time option 314
Line Print Daemon (LPD) 162
line printer
requester 160
spooler 259
load balancing FTP servers in a sysplex 108, 110
local socket 150, 165, 192, 222, 291
local-host-names file 219
LOCSITE command 85
LOCSITE parameter 373–374
LOCSTat and STAtus subcommands 90
Logical Unit (LU) 17
logmode name 333, 345
loopback file 288
LPAR 97, 108
FTP client 97
LPD Server 162, 336, 350
LPD server
simple implementation 162
LPR command 161, 163
few parameters 161
following error message 178
JES output queue 180
LPR port 178
LPR printer 166
LPR/LPD 161–162
configuration examples 163
considerations 163
dependencies 163
description 163
problem determination 176
verification 165
LU exit assignment user exit 48
LU name
master list 334
LUNAME 41

M

M4 preprocessor 214
Mail Delivery
Agent 214
mail delivery 200
mail function 198
mail message 199, 215
Mail Server 197
additional information sources 233
basic concepts 199
detailed information 233
Mail Transfer

Archived

Agent 200, 214
Management Information Base
SNMP agent communicates 130
Management Information Base (MIB) 127
managing the TN3270 server as a standalone task 43
mapping of APPL and LU assignments
design 25
mapping statement 20, 55
MBDATACONN 93
MBREQUIRELASTEOL 94
message EZZ6057I 37
MIB file 143, 327
MIB object 127
granular access control 129
textual names 149
MIB value 147
MIBDESC.DATA data set 148
MIME attachment headers 229
miscellaneous server 312, 336, 350
MONITORGROUP 142
Mozilla Thunderbird 222
MSG07 statement in the TN3270 profile 75
MTA, MUA, and MDA 214
multilevel security compliance 20
MULTIPATH PERCONNECTION 335, 349
multiple FTPD servers with SD
Stack dependencies 108
multiple TN3270
server 23, 343
multiple TN3270 server 52, 343
multiple TN3270 servers with SD
dependencies 54
description 53
stack dependencies 54
multiple TN3270 servers with Sysplex Distribution
implementation tasks 55
multiple TN3270 servers with Sysplex Distribution (SD) 52
multiple TN3270 servers, SD and GR
problem determination 77
multiple TN3270 servers, with SD
advantages 54
considerations 54
MVS data 82, 91, 322
record format 90
MVS environment 131, 198, 247

N

name server 212, 279, 281, 285
configuration file 282
only type 288
request name resolution service 285
UNIX signals 282
name server can accept queries
Verification 292
name server start procedure 288
name server started correctly
Verification 291
name server task
Start 290
named from the OMVS shell 305
NETACCESS statement 19
NETRC data 249
machine record 267
REXEC TSO client command 266
NETRC data set 266
NetSpool 169
netstat conn 267, 291
netstat display 28, 113
netstat home 264
netstat to verify server is listening on intended port 291
NETSTAT VCRT to show dynamic VIPA Connection Routing Table 64, 114
NETSTAT VDPT to show dynamic VIPA Destination Port Table 65, 114
NETSTAT VIPADCFG to show current dynamic VIPA configuration 66, 116
Netstat VIPADYN 68, 117
NETSTAT VIPADYN to show current dynamic VIPA and VIPAROUTE 68, 117
network access control (NAC) 19
Network Job Entry (NJE) 163
network node 129, 283
Network Print Facility (NPF) 161
Network Printer Manager (NPM) 169
network SLAPM2 132
network SLAPM2 subagent 143
NEWENV Function 42
NJE gateway 203
NJE gateway support 208
NJE network 163, 200
alternate domain name 208
domain name 208
SMTP servers 211
non-current profile 33
non-local mail 200
normal XCF 58, 111, 352
NSPORTADDR 53 182, 292
NVT-ASCII 87, 376

O

OBEYFILE and TESTMODE to syntax check the profile statements 33
OBEYFILE command 179
DEBUG EXCEPTION statement 73
Port 23 44
processing 33
resulting messages 34
update 33
OBEYFILE to modify a configuration 33
OBEYFILE to turn selected debug options on and off 70
OMPB issue 145
OMPROUTE configuration 59, 113
OMPROUTE started task 59, 113
OMPROUTE subagent 142
Open Shortest Path First (OSPF) 142
Open System Adapter (OSA) 157
operating Infoprint Server 171
optional user exit routine for RXSERVE 254
optionally permit the RXSERVE task to be a surrogate job

252
OSA definition 337, 351
OSA-Express Direct subagent 143
SNMP support 144
osnmp/snmp z/OS UNIX command 149
ospf_interface ip_address 339, 354

P

packet trace to capture data buffers 76
Parallel Sysplex
technology 55
PARM string 288
PARMSGROUP SSLCERTS 347
PARMSGROUP SSLPLAIN 346
PARMSGROUP UP2USER 347
partitioned data set (PDS) 87
performance monitoring data collection 51
performance statistics displays and monitoring 147
physical file structure (PFS) 76
physical interface 285
plain SSL 346
platform specific features via SITE/LOCSITE commands 85
point-to-multipoint interface 60, 341, 356
Policy Agent
configuration 329
proc 329
POOLLIMIT 0M 335, 349
popper 216
popper maildrop directory 219
PORT 23 17, 152, 333, 345
Port 992 346
Distribution Method 62
Port entry in /etc/services 100
port entry in /etc/services 242
PORT in PROFILE.TCPIP 242
port number 133, 177, 203, 242, 258
port reservation 203
port reservation 28, 133, 203, 287
list 178, 287
statement 139, 260
PORT statement 133, 203, 242
Port Statements 189
ports in PROFILE.TCPIP 190
preparing and using the NETRC data set 267
PRIMARYINTERFACE STAVIPA1LNK 337, 351
Print Interface 168
Print Services Facility (PSF) 168
printer inventory 162
definition utility 172
Manager 168
printer configuration information 169
PSF system information 171
printer inventory and Printer Inventory Manager 168
PRINTER testprt1 166
printing functions 170
problem determination 9, 194, 232–233, 235, 243, 290
procedure name 29, 134, 164, 203
process ID 3, 291
PROFILE data 257
PROFILE data set (PDS) 27
PROFILE DD
entry 31
profile statement 142
PROFILE.TCPI P 190, 285
data 247
datasets 110
file 324
PORT statement 100
Reserve ports 190
statement 220
Program Properties table attributes for TN3270 server 30
Pseudoterminals 240
PSF for z/OS (separate product) 169

Q

Queue directory 215
queue directory 219
queued session timer 50

R

RACF command 30, 134, 203
RDW option 91
RDY 65, 115
RECEC TSO using NETRC
Description 266
recfm=vb dataset from the z/OS FTP server 92
Redbooks Web site 393
Contact us xiii
remote client 168, 245, 247
optional password approach 249
orexec/rexec requests 258
orsh/rsh requests 258
print requests 168
rexec/rsh requests 251
remote client commands to support 252
remote execution 192, 245
basic concepts 245
remote execution and remote shell
Additional Information sources 277
remote host 80, 248–249
remote MTA 214
Remote Shell 245, 248
remote system 16, 92, 171, 248
Request for Comment (RFC) 159, 198
requeue 346
RESOLVER_CONFIG variable 321
RESOLVEVIA UDP 182, 292
REXEC client
command 249
program access 275–276
rexec command 247, 258
REXEC UNIX form 271
REXEC program 275
REXEC TSO 262, 264, 275
client command 262
client program 275
environment 249
request 263

REXEC TSO client command using NETRC data set 266
REXEC TSO client command using userid/password 262
REXEC TSO requests in batch including userid and password 264
REXEC TSO requests in batch without userid and password 267
REXEC TSO requests including userid and password 264
REXEC TSO requests without userid and password 266
REXEC TSO using NETRC
  dependencies 266
REXEC TSO with NETRC
  considerations 266
REXEC TSO with userid and password
  considerations 262
  dependencies 262
  description 262
REXEC UNIX client command 271
  considerations 272
  dependencies 271
  description 271
REXX support 122
RFC 822
  compliant message 228
  compliant message Compose 213
rndc command from the OMVS shell to manage the DNS daemon 306
RO SC30 62, 114
RO SC31 62, 114
root server 288
rsh command 247, 251
RSH request 251
run mode of the server 290
running FTP server
  actual jobname 100
RXSERVE cataloged procedure 253

S

SAF definitions 98
same userid 30, 134, 203
sample file 22, 170, 250, 287–288, 323
sample FTP.DATA 81
sample Infoprint Server configuration files 172
sample ISPF panels 173
SASP 295
SBDATACONN 376
SCANINTERVAL 120 29, 333, 345
second (backup) stack on the second system
  Start 61
second OMPROUTE 59, 113
second TCP/IP 55, 110
second TN3270 server on the second system
  Start 62
Secure port (SECUREPORT) 334, 348
secure sockets layer (SSL) 19
SECUREPORT 992 346
security
  type the agent should support 137
security access to JESSPOOL files 257
security server
  to define the SMTP started task 203
  to define the SNMP started task 134
sending a recfm=vb dataset from a non-z/OS client to a z/OS FTP server 92
sending all messages to a single file 3
sending messages to different files and to a single file 4
sending messages to different files based on jobname 3
sendmail and popper 213
sendmail as a client 227
sendmail configuration file 216
sendmail in the client mode 230
sendmail program 216
server 249
Server LPAR 97
Server Message Block (SMB) 171
server reply code from 250 to 226 84
Service Level Agreement (SLA) 132
Service Level Agreement (SLA) subagent 143
session management 20
Set debug and trace levels 305
SETROPTS CLASSACT 30, 135, 204
SETROPTS RACLIST 30, 134, 203, 290
setup sendmail and popper 228
setup SNMP configuration information 149
setup user MIB object information 149
shows SC30 68, 117
shows SC31 69, 118
Simple Mail Transport Protocol (SMTP) 198
Simple Network Management Protocol
  Version 2 129
Simple Network Management Protocol (SNMP) 127, 169
Simple Network Management Protocol (SNMP) subagent 169
single-byte character
  set 82, 239
SIOCSVIPA IOCTL 261, 337, 352
SMF address space name field 28
SMF records to capture TN3270 connection activity 75
SMTP domain name resolution 211
SMTP Server 199, 201, 336, 350
  JES spool 207
  mail delivery queues 231
  started task jobname 207
SMTP Server configuration 205
SMTP Server procedure JCL 204
SMTP user exit to define and filter spam mail 212
SMTPNOTE CLIST 207
SMTPNOTE CLIST for TSO support 207
SNA application 15, 169
SNMP
  additional information sources 157
  basic concepts 129
  procedure JCL 135
SNMP agent 130, 313, 336, 350
  certain features 133
  Configuration file 319
  Control file 319
  MIB variable 130
SNMP agent configuration display and management 147
SNMP client 148

Archived

SNMP manager 130, 169
  request packets 130
  security keys 147
SNMP Query Engine 139, 336, 350
SNMP Query Engine started task 148
SNMP request 139
SNMP server
  control data 136
SNMP server control data set 136
SNMP server procedure JCL 135
SNMPARMS control member for NetView 148
SNMPD.CONF file 134, 137
SNMPQE initialization and termination messages 146
SNMPv3 request 137
SSL security 29, 333, 345
stack affinity issues 28
standalone task 22–23, 32, 333, 345
  TN3270 Server Profile 29, 333
standard application 2, 16, 80, 128, 160, 186, 198, 236, 246, 280
Start INETD 221
START IUTIQDF4 338, 353
START IUTIQDF5 338, 353
START IUTIQDF6 338, 353
START OSA2080 338, 353
START OSA20A0 338, 353
START OSA20C0 338, 353
START OSA20E0 338, 353
Start sendmail 172, 216, 221
Start syslogd 7, 10
started task JCL
  system symbolics 23
started task procedure
  complete detailed listings 29
startup error messages 304
Static VIPA
  definition 337, 364
  requirement 339, 354
static VIPA
  HOME statement 260
static vipa 111, 260, 351
statistics file 215
step-by-step checklist xi
Storage Management System (SMS) 86
subagent 131
superuser ID 30, 134, 203
SYS1.PARMLIB(IKJTSOxx) member 208
SYS1.SEZA INST 247, 323
SYS1.TCPP Arm 292
SYSCLONE value 24, 332, 344
SYSFTPD DD
  card 97
  DISP 99
syslog 1, 290
syslogd 2, 4
SYSLOGD configuration file 5–6, 101, 324
Syslogd configuration file 5
  filter option 6
Syslogd facilities 5
Syslogd isolation 6
Syslogd parameters 6
syslogd to obtain early logging messages 305
Sysplex Distribution 55
  backup 55, 110
  scenario 343
Sysplex Distribution (SD) 52, 108, 333, 343
Sysplex Distributor 52, 108
  environment 108
  TN3270 servers 70
SYSPLEXPORTS DISTMETHOD BASEWLM 57, 111, 352
SYSPRINT DD SYSOUT 7, 31, 135, 164, 204, 288, 332, 344
system image 108, 200
  SMTP gateway server 200
System SC30 108, 140
System SC31 108
system symbolics 108
system symbolics for USS tables 49
SYSTSPRT DD
  JCL statement 257
  statement 256
  SYSOUT 270
  Term 256

T

Table search order (TSO) 246
task name 23, 108, 202
task procedure 28, 110, 113
TCP connection
  and/or UDP association 147
TCP CS 179
TCP MISCSRVB 336, 350
TCP OMVS
  BIND 10.20.10.241 261, 336, 350
  BIND 10.20.10.242 336, 350
  BIND 10.20.10.251 363
  BIND 10.20.10.252 363
  NOAUTOLOG NODELAYACKS 336, 350
TCP port
  20 100
  21 100
  25 220
  number 263
TCP RXSERVEB 336, 350
TCP/IP 15, 22, 79, 127, 159, 251, 280
TCP/IP host 80, 163, 197
  message exchange 197
TCP/IP network 79, 128, 163, 200–201
  electronic mail 200
  PostScript printer 164
  secure gateway 209
TCP/IP stack configuration profile 110
TCP/IP stack configuration profile data set 56
TCP/IP stack display and management 147
TCP/IP stack profile Port Reservation list 287
TCP/IP Stack redundancy 54
TCP/IP stack started task procedure 55, 110
TCP/IP subagent 142
TCPDATA configuration data set 25

Archived

TCPIP 84, 165
TCPIP profile configuration data set 133–134, 203
TCPIP.PROFILE.TCPI P 335, 349
TCPIP.SEZA INST 334, 348
TCPIP.SEZA XLD2 334, 348
TCPIPB 108, 144, 332, 344
TCPIPB.TCPP Arm 31, 99, 135, 164, 204, 332, 344
TCPIPJOBNAME statement 22
TCPIPJOBNAME TCPIPB 29, 182, 333, 345
telnet client 192
TELNET CONN displays to show TN3270 connections 62
TELNETGLOBALS block 142
  TCPIPJOBNAME statement 22
terminal capabilities database 240
terminology 247
TESTFILE.PRIN T 166
TESTMODE statement 33
TIMAFF 66, 116
time service
  Verification 194
TIMEMARK 600 29, 333, 345
TN3270 16, 77
  basic concepts 16
TN3270 connection
  activity SMF record 75
TN3270 connection activity 75
TN3270 general 352
TN3270 Payrol 57, 352
TN3270 port
  total number 35
TN3270 problem 21
TN3270 process
  very complete information 76
TN3270 profile 142
  MSG07 statement 75
  OBEYFILE command 71
  TESTMODE statement 33
  TNSACONFIG statement 142
TN3270 PROFILE data set 27
TN3270 Server 15–16, 21–22, 130, 237, 280, 316, 324, 331, 344
  Distributed DVIPA address 52
  problem determination 70
  VTAM LU definitions 23
TN3270 Server as a standalone task
  Verification 32
TN3270 Server configuration profile data set 56
TN3270 server executing as a standalone task
  advantages 28
  considerations 28
  dependencies 23
  description 22
  Implementation tasks 23
TN3270 server started task procedure 56
TN3270 servers with Sysplex Distributor
  problem determination 70
TN3270 service 20
TN3270 SNMP subagent
  activation 28
  limitation 28
TN3270 SNMP subagent limitation 28, 133
TN3270 Standalone 331
  SC30 Files 332
TN3270 Telnet subagent 142
TN3270E functionality 18
TN3270E Server 21
transfer mode 88–89, 376
transferring a recfm=vb dataset between z/OS systems 92
transferring MVS datasets to stream oriented file systems 90
Transform Manager 169
TRANSLATE 372
TRANSLATE option 376
Transport layer security
  TN3270 connections 19
Transport layer security (TLS) 19, 82
trap forwarder 132
trle definition 337, 351
TSO batch procedures for RXSERVE 256
TSO remote execution
  Considerations 251
  Description 251
TSO remote execution client 248
TSO remote execution dependencies 251
TSO remote execution environment 250
TSO remote execution server 248
TSO remote shell client 249
TSO session 236, 248, 264
  REXEC TSO client command 264
TSO user 163, 207
TSR 65, 115
TYPE118 TCPIPSTATISTICS 335, 349
TYPE119 TCPIPSTATISTICS 335, 349

U

UDP LLBD 336, 350
UDP MISCSRVB 336, 350
UDP NCPROUT 336, 350
UDP Net 350
UID 30, 134, 203
understand connection mapping terminology 26
UNICODEFILESYSTEMBOM 93
unique parameter 20, 205–206
UNIX API variable 315
UNIX environment 132, 247
update /etc/inetd.conf for popper 220
update /etc/services 220
update profile.TCPIP 220
useful to determine problems (UDP) 130
user ID 19, 81, 134, 202, 239, 321
  client workstation 239
userid 84, 134, 203, 249, 290
users access 29, 153, 334, 346
usr/sbin/rexecd rexecd 259
USS table 20, 324, 346
UTF-8 93

V

VARY ABENDTRAP to set abend traps 74
VARY DEBUG OFF to turn off all Debug options 73
VARY to change the status of TN3270 LUs 45
VARY to Quiesce, Resume, Stop a TN3270 port 43
verification 8, 11, 101, 221, 242
VIPA address 260, 284
VIPA Connection Routing Table (VCRT) 64, 114
VIPA Destination Port Table (VDPT) 65, 114
VIPA interface 59, 285, 339, 354
VIPA Range 60, 116
VIPA Route 67, 116
VIPABACKUP 200 Move 58, 112, 365
VIPADYNAMIC
  add to the backup TCP/IP stack 58
  add to the primary TCP/IP distributing stack 57
VIPADYNAMIC block 54, 109
VIPADYNAMIC statement 260
VIPAROUTE information 68, 117
Virtual Telecommunications Access Method (VTAM) 169
VTAM APPL major node 23–24
VTAM application 17, 169, 171
  data 170
  print requests 169

W

well-know port 250–251
When a simple LPD server is required for inbound print 162
When manageability of centralized printing is required 162
wildcard format 30, 134, 203
WLM 28, 115

X

XCF interface 54, 109, 341, 356
  IP subnet 56
XCF use 60, 341, 356

Z

z/OS Communications Server 2, 16, 24, 80–81, 130, 160, 170, 186, 198, 236, 246, 280, 311
  model APPL statement 24
  separate product 170
  TCP/IP subagent 142
z/OS FTP client 86, 92
z/OS FTP server 92
  vb dataset 92
z/OS SMTP
  server 199, 203
  server function 200
z/OS SMTP server
  Considerations 202
  Dependencies 202
  Implementation tasks 203
  Problem determination 231
  Verification 212
z/OS SMTP server functions
  Description 200
z/OS SNMP agent 131–133
  considerations 133
  dependencies 132
  description 132
z/OS SNMP client command 148
z/OS TCP/IP xi
z/OS UNIX 81, 167, 186, 198, 235, 246, 285, 311
  API environment 313
  API environment search order 313
  API environment variable 315
  API variable 315
  application 132, 321
  application program 240
  cat command 229
  command 236, 248, 313
  environment 131, 247, 311
  file system 4, 168, 187, 228, 240
  LP 171
  LPR command 5
  OPORTMAP 313
  OREXECD 313
  orsh client 258
  ORSHD 313
  osnmp command 138, 156
  remote execution 248
  remote execution client 248
  remote execution server 248
  remote shell client 249
  remote shell server 246
  REXEC command 272
  REXEC Server 186
  REXECD server 247
  RSH server 186
  RSHD 247
  server 236, 251
  shell 132, 167, 195, 236, 271–272, 305
  shell command line 273
  snmp command 131
  system planning effort 167
  System Service 167, 311
  System Services API 312
  System Services Environment Variable 312
  System Services kernel 313
  System Services shell 235, 248
  System Services shell issue 217
  Telnet 235
  Telnet Server 186
  Telnet server authenticates user 236
  Telnet server design scenario 235
  Telnet server function 235
  Telnet server INETD 238
  Telnet server work 236
  Telnet session 238
  z/OS UNIX remote execution servers 249
z/OS UNIX file 286
z/OS UNIX remote execution 257
  Considerations 258
  Description 258
z/OS UNIX remote execution client 248

Archived

z/OS UNIX remote execution dependencies 258
z/OS UNIX remote execution server 248
z/OS UNIX remote shell client 249
z/OS UNIX remote shell server 248
z/OS UNIX telnet server 237
z/OS V1R7.0 Communications Server 2, 79, 159, 213, 217
    product 323

Archived

Archived

CS for z/OS V1R9 TCP/IP Implementation Volume 2: Standard Applications

Archived

Archived

Archived

Communications Server for z/OS V1R9 TCP/IP Implementation Volume 2: Standard Applications

Learn about CS for z/OS TCP/IP standard applications

TCP/IP standard application implementation examples provided

Leverage TCP/IP standard applications for your needs

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

In this IBM Redbooks publication we provide useful implementation scenarios and configuration recommendations for many of the TCP/IP standard applications supported by the z/OS Communications Server.

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

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535

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

SG24-7533-00 ISBN 0738485594